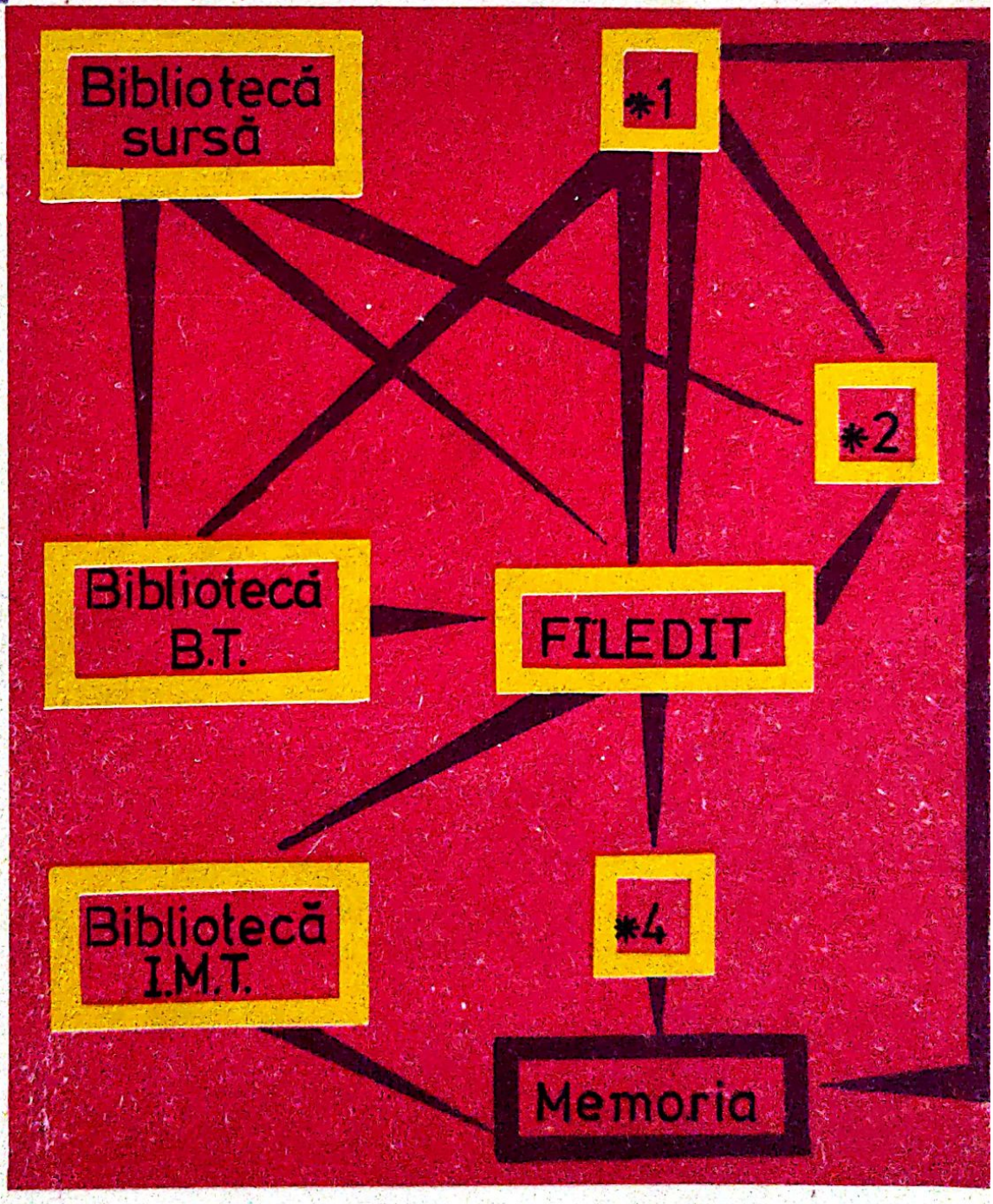
LYCEUM

COLECȚIA LYCEUM

HORIA GEORGESCU • PETRE PREOTEASA

Introducere în sistemul de operare SIRIS



EDITURA ALBATROS

Scanned with OKEN Scanner

HORIA GEORGESCU • PETRE PREOTEASA

Introducere în sistemul de operare SIRIS

Cu o prefață de acad. NICOLAE TEODORESCU

EDITURA ALBATROS • 1978

Scanned with OKEN Scanner

Acum zece ani, în 1967, organizam primul Colocviu de tehnici de calcul și calculatoare din țara noastră, prin Asociația Oamenilor de Știință din R.S. România în colaborare cu Universitatea din București — prin Centrul de Calcul al acesteia, cu participare internațională.

Informatica românească se găsea atunci la primii săi pași în domeniul tehnicii de calcul; totuși, personalități străine invitate au descoperit cu deosebită surpriză și cu admirație existența unei îmbelșugate producții de lucrări originale ale informaticienilor români, la nivelul atins de informatica internațională în acel moment.

Astăzi, producem sisteme de calcul, numărul cercetătorilor s-a înzecit, învățămîntul informaticii a fost organizat atît în învățămîntul superior universitar și tehnic, cît și în cel liceal unde avem licee de informatică.

Numărul de centre de calcul înzestrate cu calculatoarele românești FELIX C-256 crește continuu pe întreg cuprinsul țării, astfel încît necesitatea cunoașterii și stăpînirii sistemului de operare SIRIS, folosit în aceste sisteme de calcul, a devenit imperioasă pe plan național. Facultățile de matematică au secții de informatică și chiar cursuri de *Bazele informaticii* obligatorii pentru toți studenții. Cunoașterea limbajelor de programare, deși indispensabilă, nu poate da posibilitatea folosirii calculatorului FELIX C-256 fără cunoașterea sistemului de operare SIRIS de către elevi, studenți și, bineînțeles, de toți utilizatorii.

Scanned with OKEN Scanner

Sistemul de operare SIRIS dă posibilitatea de a lucra cu fișiere, de a organiza biblioteci, de a alcătui programe complexe și de a detecta erorile eventuale din acestea, de a lista și de a modifica informațiile cuprinse în diverse volume sau fișiere, precum și multe alte facilități pe care specialistul poate și trebuie să le descopere singur.

Pornind de la aceste necesități și avantaje, unul dintre cercetătorii catedrei de Informatică, dr. Horia Georgescu, împreună cu cercetătorul, binecunoscut ca autor, Petre Preoteasa de la Centrul de Calcul al Universității București, au hotărît să elaboreze prezenta lucrare intitulată „Introducere în sistemul de operare SIRIS" pe baza unei experiențe de mai mulți ani cu studenții Facultății de matematică a Universității București, experiență care se cere extinsă la un număr mult mai însemnat de utilizatori.

Pentru a preciza cadrul lucrării, vom observa că aceasta nu are ca obiect prezentarea teoriei sistemelor de operare, ci numai *sistemul de operare* **SIRIS,** într-o formă didactică, accesibilă. Este binecunoscut că documentația de firmă este dificil de însușit operativ datorită volumului mare de informații din care este alcătuită; utilizatorul trebuie să-și extragă acele informații de care are nevoie în primul rînd, ceea ce este foarte adesea dificil, atît prin alegerea și sistematizarea materialului, cît și prin faptul, esențial, că trebuie să aibă la dispoziție permanent documentația în cauză.

De aceea, lucrarea celor doi autori capătă în primul rînd meritul de a pune la dispoziția oricărui utilizator acele informații privind sistemul de operare SIRIS, care sînt cele mai utile și cele mai frecvent folosite. Importanța acestui merit decurge, fără îndoială, din faptul că economia noastră națională este înzestrată cu sistemele de calcul FELIX C-256, astfel încît se oferă oricărui utilizator principalele facilități cuprinse în sistemul de operare SIRIS. Vom sublinia încă o calitate a lucrării, aceea că, prezentînd și organizarea sistemului, dă

Scanned with OKEN Scanner

utilizatorului putința de a-și clarifica noțiunea de sistem de operare, deci de sistem de programe de bază care permite rularea programelor alcătuite de utilizator. Se știe că sistemul de operare, înscris pe un anumit disc numit disc sistem, este livrat odată cu calculatorul și că în el se pot face ulterior unele modificări care urmăresc să-l facă mai performant.

Avînd în vedere că limbajul FORTRAN este de cea mai largă circulație, fiind primul limbaj care se predă utilizatorilor cît și viitorilor informaticieni, autorii au ales acest limbaj pentru exemplificările pe care le prezintă. Ei adaugă și un capitol referitor la *depanarea erorilor* cu scopul de a permite utilizatorilor să-și corecteze singuri programele. Pentru întrebuințări curente, aceștia dispun și de un capitol în care sînt înserate programe utilitare, la care se adaugă numeroase exemple rulate pe calculator care completează capitolele lucrării servind ca modele pentru scrierea diferitelor programe care apar în activitatea obișnuită a celor ce folosesc sistemul de calcul FELIX C-256.

Experiența didactică de care dau dovadă autorii se reflectă atît în ordonarea sistematică a materialului (care este prezentat progresiv, ținîndu-se seama de cunoștințele anterior expuse), cît și în stilul folosit care este antrenant și clar. Cititorul avizat, prin consultarea documentației de firmă va ști să aprecieze acest mod didactic de inițiere în sistemul SIRIS, fiindcă, în general, documentațiile de firmă presupun cunoașterea a numeroase proceduri anterioare; de asemenea va ști să aprecieze și exemplificările care se găsesc în fiecare capitol, la locul potrivit, și într-un număr considerabil.

Scopul lucrării constă în a selecta cele mai utile instrumente de lucru și de a clarifica noțiuni de bază; nu se urmărește, în nici un caz, să se înlocuiască documentația de firmă. În conformitate cu titlul, materialul expus reprezintă numai o introducere în sistemul de operare SIRIS, cunoașterea aprofundată a acestuia impunînd un studiu mult mai prelungit și folosirea limbajului

Scanned with OKEN Scanner

cel mai adecvat, ASSIRIS, precum și a unor cunoștințe matematice de nivel ridicat.

Prin modul sistematic în care este selectat materialul, cartea se adresează elevilor din clasele speciale de informatică, studenților cărora li se predau noțiuni de programare precum și tuturor utilizatorilor sistemului de calcul FELIX C-256, din centre de calcul sau întreprinderi.

Este deci vorba de o lucrare menită să aibă o largă răspîndire și putem anticipa că va găsi numeroși cititori, aducînd atît editurii Albatros, cît și autorilor un succes remarcabil.

Acad. NICOLAE TEODORESCU

Scanned with OKEN Scanner

Capitolul I.

ORGANIZAREA GENERALĂ A SISTEMULUI FELIX

SISTEMUL DE CALCUL FELIX

Sistemul de calcul FELIX C-256 este fabricat în R.S.R. Structura și caracteristicile sale sînt asemănătoare cu cele ale calculatorului de construcție franceză IRIS-50.

Caracteristicile hardware și cele software îl situează între calculatoarele de generația a III-a, de putere medie. El poate fi utilizat pentru rezolvarea unei game largi de probleme, de la cele în care preponderent este volumul de calcule pînă la cele în care semnificativ este volumul mare de informații prelucrate.

FELIX C-256 este asemănător din punct de vedere constructiv (hardware) cu calculatoarele din aceeași perioadă cu putere de calcul similară (de exemplu cu calculatoarele IBM din seria 360, Siemens etc.). Regăsim în acest sistem de calcul principalele părți care intră în componența unui calculator modern și anume:

— memoria, pentru păstrarea informațiilor;
— unitatea de comandă și calcul;
— unități de schimburi (canale pentru a facilita transferul datelor între diferitele unități ale sistemului și memorie);
— unități de legătură;
— dispozitive periferice.

Această împărțire a sistemului de calcul în subsisteme independente, cu funcții bine definite în cadrul sistemului, asigură calculatorului FELIX C-256 (și de altfel acest lucru este valabil pentru orice calculator similar) următoarele avantaje:

a) *O funcționare simultană* a unor subansambluri.

b) *Modularitate*, în sensul că, după necesitățile utilizatorului, poate fi realizată o configurație corespunzătoare.

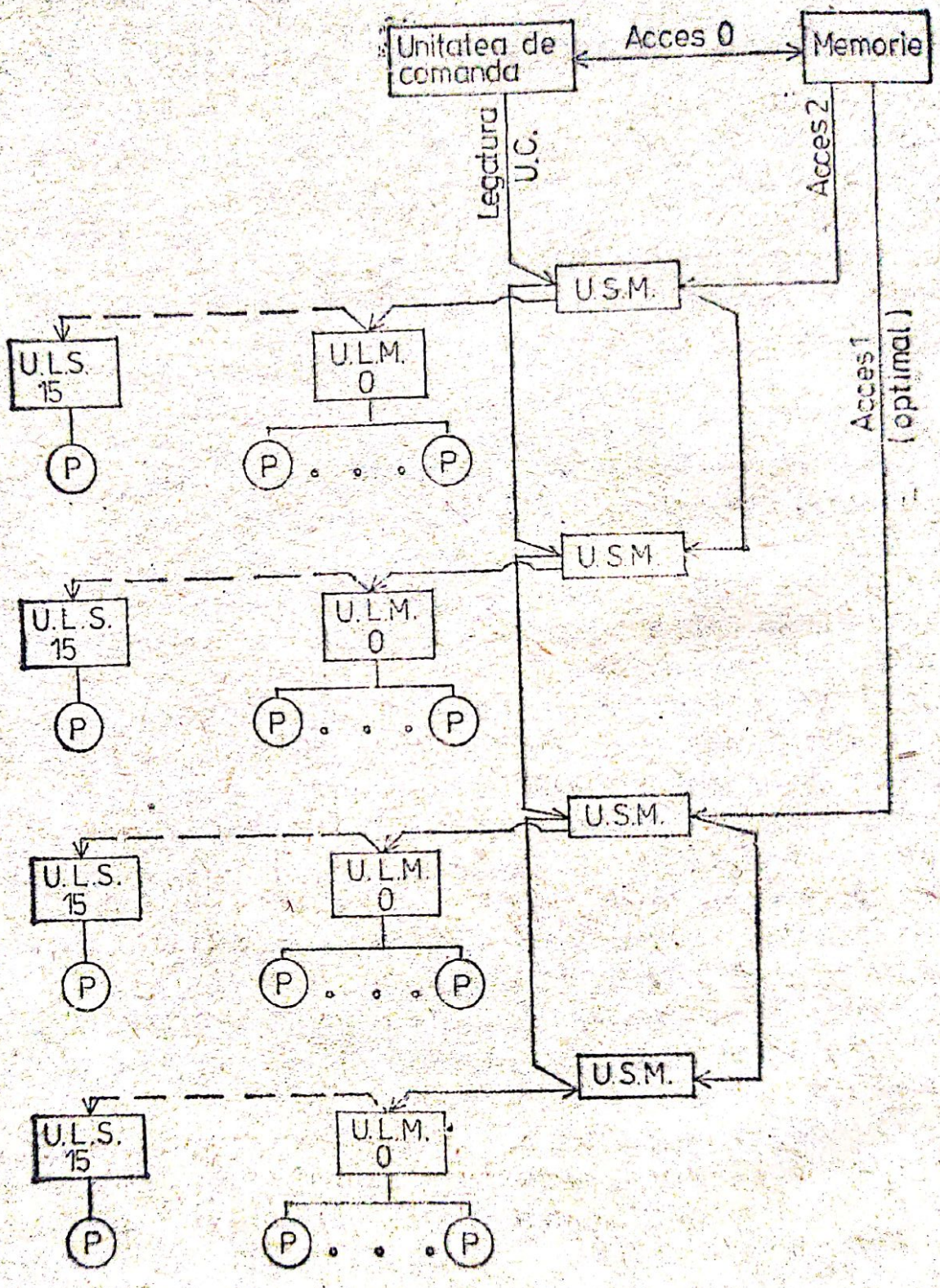


Fig.1 Structura sistemului de calcul FELIX.

Scanned with OKEN Scanner

Din punct de vedere al posibilităților de programare oferite utilizatorului (software), sistemul de operare, deși propriu acestui sistem de calcul, este similar cu al altor calculatoare din aceeași categorie.

Regăsim astfel în sistemul de operare SIRIS componentele de bază ale altor sisteme de operare.

Limbajele de programare pe care sistemul de operare SIRIS le pune la dispoziție utilizatorului sînt variate, permițînd abordarea problemelor de natură tehnică și științifică, precum și a celor de gestiune în timp real sau de teleprelucrare.

UNITATEA DE MEMORIE

În cursul unor prelucrări calculatorul operează asupra datelor care pot fi, după natura prelucrării, de diferite feluri. Utilizatorul este interesat în mod direct de următoarele două tipuri de date:

— date de intrare, furnizate ca date inițiale pentru calculele ce se vor efectua;

— date de ieșire sau rezultate obținute după o prelucrare.

În afară de acestea în cursul procesului de calcul apar și sînt folosite și alte date care pot fi rezultate intermediare ale unor calcule, indicatori de stare ai unor operații etc. Ele nu sînt vizibile direct pentru utilizator care, în principiu, nu este interesat de acestea decît în măsura în care nu obține rezultate corecte la problema sa.

Pentru a păstra aceste date și pentru a le utiliza, sistemul de calcul trebuie să dispună de dispozitive unde ele să fie păstrate și eventual folosite. Aceste dispozitive alcătuiesc memoria calculatorului.

În general, nu toată memoria unui calculator are aceleași caracteristici. În calculatoarele din generația actuală se desprind cel puțin două nivele. La nivelul 1, o memorie principală sau centrală care asigură un acces direct la date, o perioadă de citire sau scriere a datelor foarte mică (de

Scanned with OKEN Scanner

ordinul microsecundelor sau nanosecundelor) și la nivelul 2, o memorie auxiliară în care accesul se face mai lent (de ordinul milisecundelor sau mai mult).

Memoria principală este relativ scumpă în raport cu memoria auxiliară. De aceea sînt păstrate în memoria principală pe timpul efectuării unei lucrări numai datele strict necesare, restul fiind aduse la cerere anterior utilizării lor.

Constructiv, memoria principală este alcătuită la calculatorul FELIX C-256, din inele de ferită aranjate conform unei structuri specifice, fiecare din acestea permițînd păstrarea unei cifre binare: 0 sau 1 (prin inele sînt trecute fire în care circulă curent electric; în funcție de sensul și valorile curenților rezultă două sensuri de magnetizare care sînt puse în corespondență cu cifrele binare 0 și 1).

Cel mai mic element constitutiv al memoriei este *bit*-ul (prescurtare de la *binary digit* = cifră binară); după cum arată numele său, un bit poate memora o cifră binară.

Opt biți consecutivi dintre care primul are numărul de ordine divizibil cu opt poartă numele de *octet* (bait).

Memoria internă poate fi imaginată ca lineară, fiecare octet din memorie avînd un număr de ordine numit *adresă* (primul octet are numărul de ordine = 0).

Capacitatea memoriei interne exprimă numărul octeților care o alcătuiesc.

Un cuvînt este o succesiune de patru octeți consecutivi astfel încît adresa primului este divizibilă cu 4.

Datele sînt citite și scrise în memorie. La FELIX C-256 într-o singură operație de acest gen sînt citiți (scriși) doi octeți. Pentru a avea acces la memorie pentru astfel de operații este nevoie de codul operației (citire sau scriere), adresa informațiilor și lungimea lor.

Accesul pe doi octeți la memorie implică anumite restricții de adresare. Astfel adresa unui semicuvînt trebuie să fie multiplu de doi, a unui cuvînt multiplu de patru iar a unui dublu cuvînt, multiplu de 8. Deci în funcție de tipul operației ce va fi efectuată pe o anumită zonă de memorie trebuie să avem o adresă corespunzătoare.

STRUCTURA MEMORIEI

Memoria calculatorului FELIX C-256 are o structură modulară, în sensul că este posibil ca unui sistem de calcul dat să-i fie adăugată ușor memorie suplimentară. Memoria calculatorului se măsoară în kilocteți (notație: koctet).

Un kiloctet este format din 1024 octeți. Capacitatea memoriei centrale este un criteriu important pentru măsurarea performanțelor unui calculator, deoarece accesul la orice informație din memoria internă este mult mai rapid decît accesul la o informație aflate pe un periferic și cu cît memoria internă este mai mare, cu atît mai ușoară este executarea programelor de dimensiuni mari (am văzut că rularea unui program presupune existența sa sau a părții din care face parte instrucțiunea ce se execută în acel moment, în memoria centrală).

Teoretic memoria calculatorului FELIX poate varia între 16 kocteți și 256 kocteți. În practică sînt mai puține configurațiile în care memoria este mai mică de 64 kocteți (de fapt utilizarea sistemului standard de operare SIRIS al calculatorului limitează el însuși limita inferioară datorită dimensiunii programelor sale).

Memoria internă este organizată modular. Unitatea independentă de memorie este *blocul de memorie*. Un bloc de memorie este format din subansamble de cîte 16 kocteți numite *module*. Memoria internă poate fi formată din 1—4 blocuri de memorie numerotate 0,1,2,3. Dacă există un singur bloc de memorie, el poate fi format din 1,2,3 sau 4 module. Dacă există mai multe blocuri de memorie, numai ultimul poate fi eventual format din mai puțin de 4 module, blocurile anterioare conținînd exact 4 module.

Opțional memoria posedă un dispozitiv de protecție al memoriei. În acest caz fiecare 2 kocteți (numiți și pagină) posedă o cheie de protecție formată din 4 biți.

Orice acces la memorie este însoțit de o *cheie de acces*. În cazul în care cheia de acces este egală cu cheia de protecție operația este efectuată. Contrar, operația este efectu-

ată numai în anumite cazuri bine precizate, care depind de relația ce există între cheia de acces și cheia de protecție.

Existența dispozitivului de protecție a memoriei este obligatorie cînd în memoria centrală sînt mai multe programe care se execută în același timp (multiprogramare) și cînd nu este permis unui program să deterioreze datele din zona altui program.

În unitatea de memorie centrală sînt păstrate instrucțiunile, operanzii și comenzile de intrare/ieșire ale unui program. Trebuie spus de la început că orice program, pentru a fi executat, trebuie să fie adus în memoria centrală (internă). Programul aflat în memoria internă este format din instrucțiuni în cod-mașină, adică instrucțiuni direct executabile de către calculator.

De aceea la memorie au acces:

— unitatea de comandă;

— unitățile de schimburi multiple.

Datele schimbate în cursul prelucrării între aceste unități și memorie sînt transmise pe căi speciale (interfață magistrală).

Între memoria internă și celelalte componente ale sistemului pot exista maximum trei căi de comunicație (una cu unitatea de comandă și una sau două cu unitățile de schimburi multiple).

UNITATEA DE COMANDĂ

Prelucrările executate de un sistem de calcul constau în ultimă instanță dintr-un șir de operații elementare, fiecare dintre acestea traducîndu-se în limbajul calculatorului printr-o instrucțiune în cod mașină.

Unitatea centrală este acea parte a sistemului de calcul care va îndeplini analiza, execuția și înlănțuirea acestor instrucțiuni (primind o instrucțiune, unitatea centrală stabilește despre ce operație este vorba, care sînt operanzii și îndeplinește operația indicată; tot unitatea centrală va deter-

Scanned with OKEN Scanner

mina adresa următoarei instrucțiuni de analizat și executat reluînd ciclul).

Calculatorul FELIX C-256 are instrucțiuni de lungime fixă (4 octeți). În fiecare instrucțiune este indicată printr-un cod operația de efectuat și anumite date care în final duc la determinarea adresei operanzilor sau chiar a operanzilor.

Observații

Numai rareori utilizatorul scrie programe direct în cod-mașină. De cele mai multe ori el folosește limbaje de programare, transformarea programelor scrise în aceste limbaje într-un program în cod-mașină făcîndu-se automat de către un compilator sub controlul unui sistem de operare.

Pentru a-și îndeplini acest rol de conducător al efectuării prelucrării, unitatea centrală duce un dialog continuu cu unitatea de memorie (deoarece operanzii ca și instrucțiunile sînt în general în memoria centrală) și cu unitățile de schimburi multiple (deoarece anumite date care nu sînt în memoria centrală trebuie aduse în aceasta din memoriile auxiliare înainte ca unitatea centrală să treacă la folosirea lor).

Operațiile efectuate de unitatea de comandă sînt complexe. Pentru îndeplinirea lor se disting în cadrul ei mai multe părți, fiecare din ele avînd funcțiuni bine precizate.

— 16 registre generale;
— registre auxiliare;
— operatorul binar;
— operatorul de calcul zecimal;
— operatorul de calcul în virgulă mobilă;
— panoul de comandă;
— sistemul de tratare a devierilor;
— sisteme de tratare a întreruperilor;
— orologiu;
— legătura programată;
— pupitrul de comandă.

a) *Registrele generale.* Calculatorul FELIX C-256 dispune de 16 registre generale numerotate de la 0 la 15 a

Scanned with OKEN Scanner

cîte 32 biți fiecare. Ele dublează din punct de vedere logic primii 64 octeți ai memoriei (deci au adresele 0,4,8.....60).

Un registru general poate îndeplini diferite funcții în cursul prelucrării: memorie intermediară rapidă (poate păstra un operand, rezultatul unei operații sau lungimea unui operand), mijloc pentru identificarea operanzilor (registru de bază, index).

Din punct de vedere constructiv aceste registre (ca de altfel și registrele auxiliare) sînt realizate folosind bistabili (spre deosebire de memorie care folosește pentru memorare inele de ferită).

b) *Registrele auxiliare* (se mai numesc registre asociate sau registre interne).

În funcționarea sa pentru prelucrarea unor date, unitatea de comandă utilizează mai multe registre auxiliare care în general nu sînt accesibile programatorului. Operația elementară (exemplu, adunarea conținutului a două adrese din memorie care constituie cei doi operanzi) presupune la rîndul ei o descompunere mai fină. Informativ putem distinge următoarele etape:

— analiza instrucțiunii mașină;

— găsirea adresei operanzilor;

— stabilirea operației ce trebuie efectuată (decodificare);

— aducerea operanzilor în registrele auxiliare (după cum am văzut în cazul unui acces la memorie pot fi scriși sau citiți doi octeți; deci pentru operanzii cu o lungime mai mare de 2 octeți vor fi necesare mai multe cereri de acces la memorie pentru obținerea lor);

— efectuarea operației în operatorul corespunzător (aceasta la rîndul ei poate fi îndeplinită în mai multe etape obținîndu-se în fiecare dintre ele un rezultat parțial);

— transferul rezultatului din operatorul corespunzător în memorie dacă este cazul.

Dintre funcțiile pe care le au registrele auxiliare putem distinge:

— constituie intrările și ieșirile pentru operatorii unității de calcul (binar, virgulă mobilă, zecimal);

Scanned with OKEN Scanner

— sînt utilizate pentru adresarea memoriei (conțin adresa de la care se va face citirea sau la care se va face scrierea);

— conțin informația ce va fi scrisă în memorie;

— conțin informațiile citite din memorie la încheierea unei asemenea operații;

— participă la decodificarea codului instrucțiunii;

— pentru vizualizarea unor informații pe panoul de comandă;

— pentru păstrarea unor informații legate de înlănțuirea instrucțiunilor (adresa următoarei instrucțiuni, cheia de acces, măștile de depășire etc.);

— sînt folosite în cursul efectuării operațiilor de intrare-ieșire.

Am dat aceste detalii pentru a-l face pe cititor să realizeze următorul fapt: scriind în limbaj-mașină, programatorul descrie operațiile ce le va executa calculatorul, operații elementare din punctul lui de vedere. Aceste operații vor fi descompuse la rîndul lor în operații elementare din punctul de vedere al unității de calcul; acestea din urmă sînt cele care vor fi efectiv executate.

Registrele auxiliare folosite de U.C. sînt A,B,M, I,J, S,G, P,D,E,EP.

Pentru noi o importanță mai mare va prezenta (chiar dacă indirect) conținutul registrelor EP și P care conțin un grup de informații ce descriu condițiile în care este executat un program. Aceste informații le vom desemna sub numele *starea program*. Ele sînt conținute în așa numitul *dublu cuvînt de stare a programului*. Accesul programatorului la unele informații conținute în acest cuvînt e limitat și posibil numai utilizînd limbajul ASSIRIS. Totuși, cunoașterea structurii lui este utilă pentru depanarea programelor.

Forma acestui dublu cuvînt este dată în fig. 2.

Dintre informațiile înscrise în acest cuvînt dublu de stare menționăm următoarele:

— *biții 14—31* conțin adresa program. Aici este memorată adresa următoarei instrucțiuni care trebuie executată, calculată diferit după cum instrucțiunea curentă este o instrucțiune de salt sau nu.

— *biții 32—35* reprezintă măști de depășire pentru operații binare, zecimale sau în virgulă mobilă. La apariția unei depășiri se produce o deviere numai dacă bitul corespunzător este 0; în caz contrar depășirea este ignorată. În mod normal acești biți au valoarea zero; după cum am menționat, această valoare poate fi modificată printr-un program scris în ASSIRIS.

— *biții 40—42* reprezintă măști de întrerupere. Acestea sînt asemănătoare cu cele de depășire. Cu ajutorul lor se pot masca întreruperi externe sau interne.

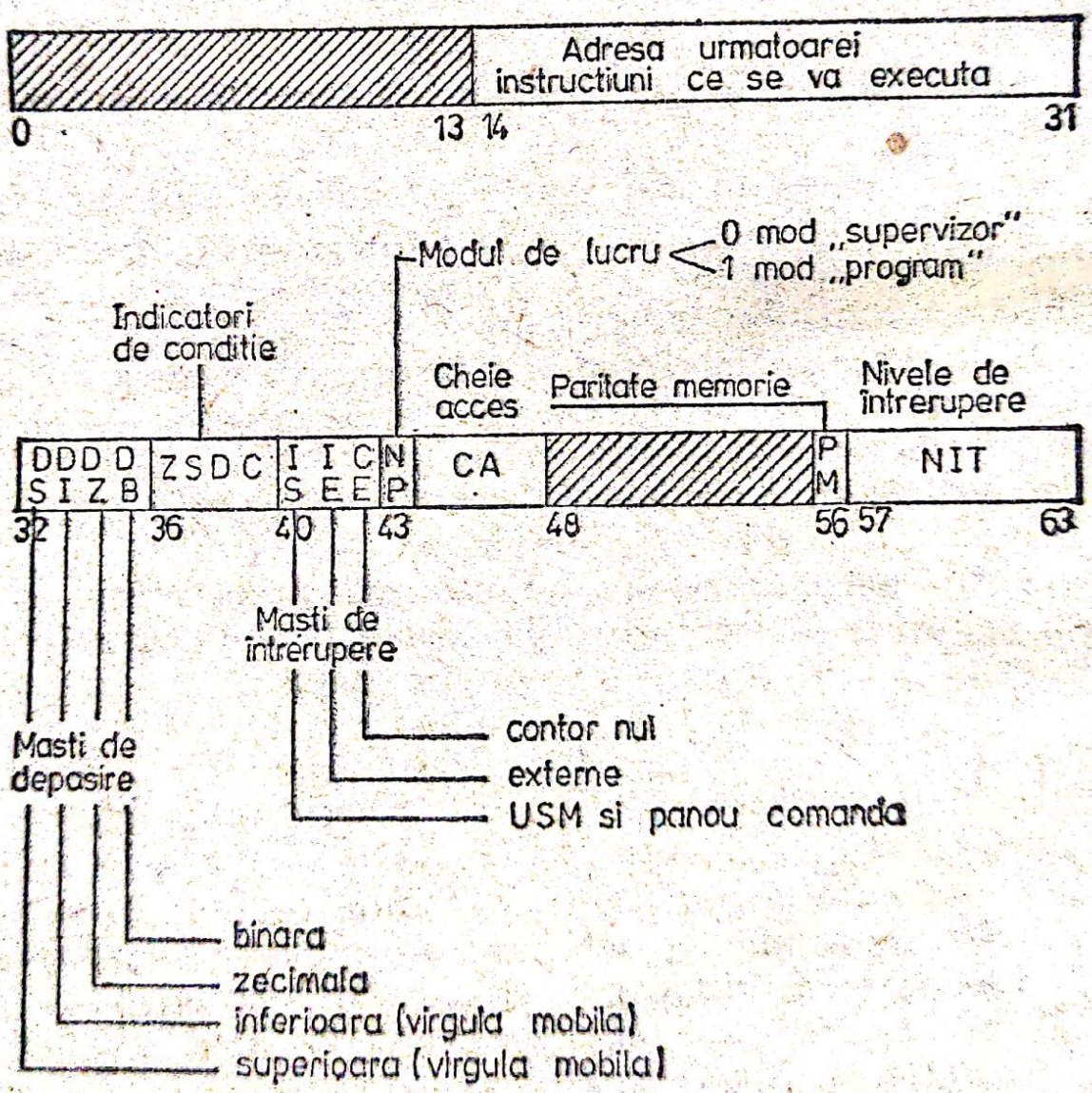


Fig.2 Dublul cuvînt de stare al programului

— *biții 44—47* formează cheia de acces care în funcție de rezultatul comparării cu cheia de protecție a unei zone din memorie permite scrierea și citirea din zona respectivă a unor informații.

— *biții 57—63*, reprezintă nivelul de întrerupere. Acesta definește prioritatea programului în curs de execuție. Pentru ca acest program să poată fi întrerupt, nivelul care cere întreruperea trebuie să fie mai mic (deci mai prioritar) decît cel al programului ce se execută.

— *bitul 43*. Dacă acest bit are valoarea 0 atunci în programul corespunzător acestui dublu cuvînt de stare poate fi folosită orice instrucțiune din setul calculatorului FELIX C-256. În schimb, dacă valoarea este egală cu 1 încercarea de a executa o instrucțiune privilegiată provoacă o deviere: „violare de mod". În general programele scrise de către programatori au acest bit poziționat pe 1.

— *biții 36—39*. După efectuarea anumitor operații, valorile acestor biți indică relația care a existat între operanzi și anumite situații care pot apare (de exemplu depășire).

c) Executarea instrucțiunilor cade în sarcina celor 3 operatori de calcul existenți în configurația unui sistem de calcul FELIX C-256: operatorul binar, operatorul de calcul zecimal, operatorul de calcul în virgulă mobilă.

Operatorul binar are ca rol efectuarea operațiilor aritmetice asupra datelor reprezentate binar sau a operațiilor logice asupra datelor logice.

Operatorul de calcul zecimal. Permite efectuarea de operații aritmetice asupra datelor reprezentate zecimal.

Operatorul de calcul în virgulă mobilă asigură executarea operațiilor aritmetice pentru care operanzii sînt reprezentați intern în virgulă mobilă. Datorită modularității sistemului FELIX este necesară numai prezența operatorului binar. Chiar dacă operatorul în virgulă mobilă sau operatorul zecimal nu există, pot fi executate programe care utilizează acest tip de date cu condiția ca în sistemul de operare să fie înglobate rutine care, utilizînd operațiile executate de operatorul binar, să simuleze operațiile ce ar trebui

Scanned with OKEN Scanner

executate de operatorul absent. Bineînțeles aceasta va duce la scăderea performanțelor.

d) *Panoul de comandă* apare pe partea din față a unității de comandă. Pe acest panou sînt situate chei pentru indicarea unor comenzi manuale precum și mijloace de vizualizare prin a căror utilizare este posibilă pornirea, exploatarea, supravegherea și oprirea calculatorului. În unitatea de comandă se află imaginea registrului care conține adresa următoarei instrucțiuni care trebuie executată. Această adresă se obține fie din adresa înscrisă precedent prin mărirea cu 4 (o instrucțiune ocupînd întotdeauna 4 octeți) în cazul cînd nu este vorba de o instrucțiune de salt, fie din informația din instrucțiunea curentă, în cazul în care aceasta este o instrucțiune de salt.

e) *Pupitrul de comandă* este o componentă opțională a calculatorului. Pe panoul de comandă există o cheie (LOCAL/OPERATIONAL) a cărei poziție arată dacă sistemul a luat sau nu în considerație comenzile ce vin de la pupitrul de comandă. Pe aceasta din urmă sînt cîteva butoane care dublează comenzile cele mai utilizate într-o folosire normală a calculatorului (cerere de întrerupere, comenzile necesare pentru inițializarea sistemului etc.).

f) *Sistemul de legături programate* permite conectarea directă a unor periferice, lucru care se realizează prin instrucțiunile directe de intrare/ieșire de care dispune unitatea de comandă.

g) *Orologiul*. Pentru buna funcționare a U.C. este necesară o sincronizare a efectuării operațiilor elementare care în final vor duce la îndeplinirea acelor operații cerute de utilizator. Această sincronizare se face pe baza unui semnal periodic eliberat de un generator de tact numit orologiu. Perioada acestui semnal poate fi 250 ns sau 350 ns. Orice bistabil al U.C. nu-și poate schimba starea sa (trece din 1 în 0 sau din 0 în 1) decît în momentul apariției acestui semnal de sincronizare. Pentru a avea o bună funcționare a U.C. acest semnal ajunge practic în același moment în

Scanned with OKEN Scanner

orice punct al U.C. După cum am mai spus în paragraful despre registre auxiliare, orice operație se descompune în operații elementare. Aceste operații elementare (transferul unor informații dintr-un registru în altul, o operație de adunare a operatorului binar etc.) se desfășoară între două semnale ale orologiului.

h) *Sistemul de tratarea întreruperilor.* În cursul funcționării U.C. pot apare situații de excepție. Fiecare din acestea trebuie analizată și luată o decizie corespunzătoare. Pentru fiecare asemenea situație excepțională este generat un semnal de întrerupere. Calculatorul FELIX C-256 are posibilitatea de a avea un sistem de întreruperi împărțit în 6 grupe, fiecare avînd 16 nivele (în total 96 nivele), numerotate de la 32 — 127. Totuși, în mod obișnuit numai nivelele de întrerupere din prima grupă sînt prezente.

Pentru fiecare întrerupere prezentă există un program specific de tratare a lui. Execuția acestui program este condiționată de:

— apariția semnalului de întrerupere;

— o serie de verificări ca: valoare NIT din cuvîntul dublu de stare a programului ce se execută în acel moment, valoarea măștii corespunzătoare acestui tip de întrerupere (dacă ea se poate masca).

În esență tratarea unei întreruperi înseamnă lansarea programului corespunzător ei. Acest lucru se face punînd în registrele P și EP informațiile referitoare la acest program (noua stare program) după ce în prealabil au fost memorate într-o zonă din memorie informațiile referitoare la programul în curs de execuție (vechea stare program).

Să analizăm cum se petrec lucrurile în momentul în care se produce o singură întrerupere.

În această situație se mai folosește încă un cuvînt dublu de stare corespunzător întreruperii. Acesta înlocuiește pe cel curent devenind noua stare program; cuvîntul dublu curent devine vechea stare program. Lucrurile decurg în continuare conform noii stări program. În acest nou cuvînt dublu de stare, adresa program conține locul din memorie unde se află rutina de rezolvare a întreruperii sau devierii;

Scanned with OKEN Scanner

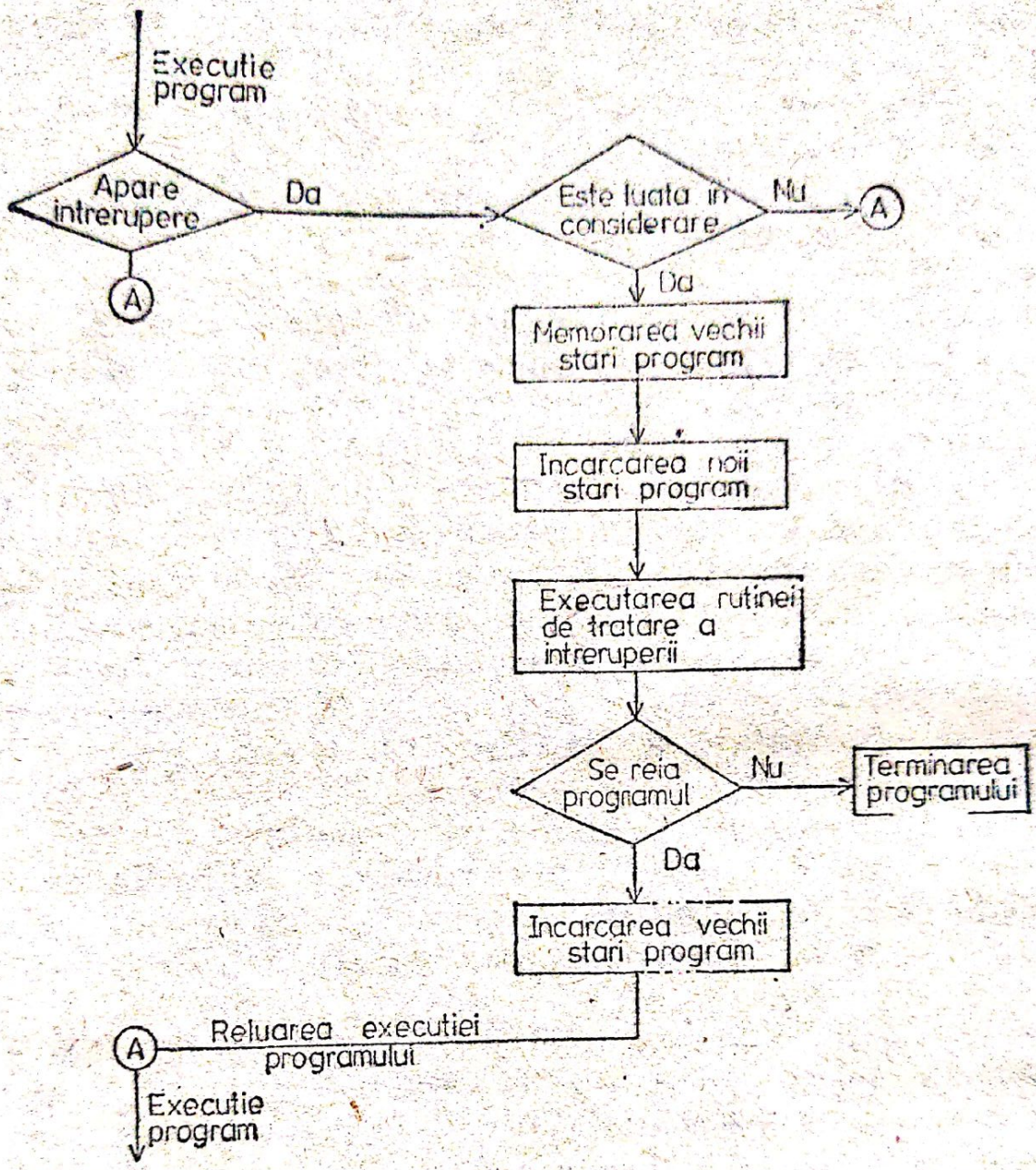


Fig 3 Modul de tratare a întreruperilor

deci această rutină este executată în continuare. În momentul în care executarea rutinei este terminată și nu a mai intervenit nici o altă întrerupere, vechea stare program revine ca stare curentă și deci programul se reia de unde a fost suspendată execütarea lui (menționăm însă că o rutină de rezolvare a unei întreruperi poate determina terminarea programului). Aceste schimbări ale stărilor program precum și eventuala reluare a programului sînt făcute de către

Scanned with OKEN Scanner

sistem fără intervenția operatorului sau a programatorului.

Întreruperile se împart în două categorii: întreruperi interne și întreruperi externe.

Exemple de întreruperi interne sînt următoarele:

— operatorul apasă pe tasta de întrerupere de la panoul de comandă;

— apare un defect al memoriei interne

— la o operație de intrare/ieșire apare un incident care împiedică desfășurarea normală a operației;

— se produce o scădere bruscă de tensiune care poate duce la perturbarea funcționării circuitelor calculatorului.

Întreruperile externe pot fi provocate de dispozitivele periferice conectate la calculator. În cazul apariției simultane a mai multor întreruperi este luată în considerație cea cu prioritate mai mare, celelalte fiind puse în așteptare.

i) *Sistemul de tratare a devierilor*

Cînd situația de excepție este generată de o instrucțiune a programului în curs de execuție are loc o *deviere* care suspendă executarea programului pînă la rezolvarea cauzei care a produs-o.

Menționăm următoarele categorii de devieri:

— devieri „instrucțiune inexistentă" — în program este întîlnită o instrucțiune care nu este una dintre instrucțiunile cablate în unitatea de comandă.

— devieri „excepție program". Ele pot fi:

- depășire binară
- depășire zecimală
- depășire inferioară în virgulă mobilă
- depășire superioară în virgulă mobilă
- încercarea de execuție a unei instrucțiuni privilegiate de către un program pentru care NP = 1 (bitul 43 din dublul cuvînt de stare a programului)
- referirea la o adresă inexistentă în memorie
- încercarea de acces la o zonă protejată a memoriei
- orologiul de gardă — are loc cînd intervalul de timp dintre două puncte „interuptibile" a depășit o anumită valoare

Scanned with OKEN Scanner

— devieri „eroare paritate". Aceasta are loc cînd s-a detectat o eroare de paritate și bitul 56 din dublul cuvînt de stare a programului este 0. Dacă acest bit este poziționat pe 1 are loc o întrerupere „defect memorie".

Depășirile de orice tip apar în special din erori de programare: împărțiri cu 0, înmulțirea a două numere prea mari, etc. Aceste depășiri pot fi mascate de către programatori (folosind limbajul ASSIRIS) punînd 1 în bitul corespunzător depășirii (32—35) al dublului cuvînt de stare a programului.

Celelalte tipuri de devieri nu pot fi mascate (de exemplu tentativele de execuție a unor operații interzise sau inexistente).

Tratarea devierilor este în mare asemănătoare cu cea a întreruperilor. Pentru fiecare deviere există o rutină în calculator care este activată pentru devierea respectivă. Întreruperile și devierile au o mare importanță pentru execuția unui program deoarece apariția și tratarea lor au loc în timpul acestei execuții.

Un program aflat în memoria internă a calculatorului poate să se găsească în două stări: activ sau inactiv.

Spunem că programul se află în *stare activă* dacă instrucțiunile sale sînt în curs de prelucrare de către unitatea de comandă. Dacă unitatea de comandă nu execută instrucțiuni ale programului spunem că aceasta este *în stare inactivă*.

Un program este în stare inactivă într-una din următoarele situații:

— este în așteptarea unui eveniment extern

— este întrerupt

Un exemplu de caz în care programul este în așteptarea unui eveniment extern este acela cînd programul citește niște date iar instrucțiunea următoare folosește aceste date. În acest caz, deși unitatea de schimburi multiple este aceea

care preia sarcina executării operației de intrare, unitatea de comandă nu poate trece la executarea următoarei instrucțiuni pînă cînd operația de intrare nu a fost încheiată.

UNITĂȚILE DE SCHIMBURI MULTIPLE

Aceste unități sînt concepute în principal pentru a permite funcționarea unității de comandă simultan cu funcționarea perifericelor.

Unitatea de schimburi este un ansamblu logic la care unitatea de comandă se adresează ori de cîte ori trebuie executată o operație de intrare/ieșire. Ea primește toate informațiile necesare executării operației respective și comandă perifericelor executarea ei, eliberînd unitatea de comandă de această sarcină; astfel, în timp ce se execută operația de intrare/ieșire, unitatea de comandă poate executa alte instrucțiuni de program.

La unitatea de memorie pot fi conectate cel mult patru unități de schimburi prin intermediul unor canale de acces. La un canal de acces la memorie sînt conectate cîte două unități de schimburi.

Fiecare unitate de schimburi multiple este conectată la periferice prin intermediul unor unități de legătură.

Există două tipuri de unități de legătură:

— *unități de legătură simple* la care poate fi conectat un singur periferic;

— *unități de legătură multiple* la care pot fi conectate pînă la 16 periferice.

La o unitate de schimburi se pot conecta maximum 32 unități de legătură simple sau 8 unități de legătură multiple. La aceeași unitate de schimburi pot fi conectate atît unități de legătură simplă, cît și multiplă.

O unitate de schimburi multiple poate fi imaginată ca un mic calculator deoarece are memorie proprie, are capacitatea de a executa operații (de intrare-ieșire) și are orologiu propriu.

Scanned with OKEN Scanner

care preia sarcina executării operației de intrare, unitatea de comandă nu poate trece la executarea următoarei instrucțiuni pînă cînd operația de intrare nu a fost încheiată.

UNITĂȚILE DE SCHIMBURI MULTIPLE

Aceste unități sînt concepute în principal pentru a permite funcționarea unității de comandă simultan cu funcționarea perifericelor.

Unitatea de schimburi este un ansamblu logic la care unitatea de comandă se adresează ori de cîte ori trebuie executată o operație de intrare/ieșire. Ea primește toate informațiile necesare executării operației respective și comandă perifericelor executarea ei, eliberînd unitatea de comandă de această sarcină; astfel, în timp ce se execută operația de intrare/ieșire, unitatea de comandă poate executa alte instrucțiuni de program.

La unitatea de memorie pot fi conectate cel mult patru unități de schimburi prin intermediul unor canale de acces. La un canal de acces la memorie sînt conectate cîte două unități de schimburi.

Fiecare unitate de schimburi multiple este conectată la periferice prin intermediul unor unități de legătură.

Există două tipuri de unități de legătură:

— *unități de legătură simple* la care poate fi conectat un singur periferic;

— *unități de legătură multiple* la care pot fi conectate pînă la 16 periferice.

La o unitate de schimburi se pot conecta maximum 32 unități de legătură simple sau 8 unități de legătură multiple. La aceeași unitate de schimburi pot fi conectate atît unități de legătură simplă, cît și multiplă.

O unitate de schimburi multiple poate fi imaginată ca un mic calculator deoarece are memorie proprie, are capacitatea de a executa operații (de intrare-ieșire) și are orologiu propriu.

Scanned with OKEN Scanner

Funcțiile lor principale sînt următoarele:

— permit schimbul de informații între utilizator și sistemul de calcul;

— joacă rolul de memorii auxiliare. Pe unele periferice poate fi înmagazinată o cantitate de informații mult mai mare decît cea care poate apare în memoria internă; în plus, ele sînt mult mai ieftine decît memoria internă.

— permit dialogul dintre operator și sistemul de calcul astfel încît operatorul poate supraveghea funcționarea calculatorului și poate lua decizii asupra modului de continuarea a unui program.

În continuare descriem principalele unități periferice cu care sînt echipate calculatoarele FELIX C-256.

1) *Cititorul de cartele* permite citirea cartelelor perforate cu 80 de coloane indiferent de principiul constructiv pe care-l are la bază. Pentru majoritatea cititoarelor de cartele folosite la calculatoarele FELIX fiecare cartelă este citită în ordinea coloanelor printr-un procedeu fotoelectric; concomitent are loc și trecerea fiecărui caracter în reprezentarea internă.

Există aici mai multe coduri de reprezentare internă și de perforare a cartelelor. Cel mai folosit este codul EBCDIC.

Conform acestui cod fiecare caracter este reprezentat intern printr-o succesiune de 8 cifre binare; de asemenea fiecărui caracter îi corespunde un sistem de perforații pe o coloană a unei cartele. Acest cod este prezentat în anexa 1.

Menționăm că în memoria internă există o *zonă tampon* care ocupă 80 de octeți, corespunzător celor 80 de coloane ale unei cartele. La citire în tampon este înscrisă imaginea cartelei în cod intern; informația respectivă este apoi interpretată conform indicațiilor date de utilizator în instrucțiunea de intrare.

La citire se fac două verificări. Prima dintre ele constă în verificarea corectitudinii citirii și se realizează prin existența a două nivele de citire conținînd fiecare 12 celule foto-

electrice corespunzătoare celor 12 linii ale cartelei; citirea este validată numai dacă rezultatele celor două citiri coincid. A doua verificare constă în controlul validității perforațiilor; caracterele cu perforații greșite sînt introduse într-o magazie de recepție secundară.

Oprirea citirii poate avea loc numai după citirea completă a unei cartele. Viteza maximă de citire este de 1200 caractere/min.

2) *Perforatorul de cartele*

Sub acest nume sînt cunoscute două tipuri distincte de perforatoare de cartele. Primul servește la pregătirea manuală a cartelelor pe care este înscris programul utilizatorului. Acest tip de perforator nu este conectat la calculator. Perforarea se face coloană cu coloană.

Al doilea tip de perforator de cartele este conectat la calculator și servește la scoaterea rezultatelor programului pe cartelă. Debitul său maxim de 300 cartele/min. Intreruperea perforării se poate face numai după perforarea unei cartele complete. Acest tip de perforator este prevăzut cu un sistem de verificare. La perforare, datele care trebuie perforate sînt întîi înscrise în zona tampon și apoi, conform codului folosit, sînt perforate pe cartele.

3) *Imprimanta* este un periferic frecvent utilizat pentru scoaterea rezultatelor. La imprimantă se pot obține de asemenea liste ale programelor, conținutul memoriei, al fișierelor sau bibliotecilor, mesaje asupra erorilor, etc.

Imprimanta utilizată de sistemul FELIX imprimă datele linie după linie cu viteză maximă de 800 linii pe minut. O linie a hîrtiei de imprimantă conține 132 de caractere. Spațierea pe verticală a imprimării datelor se realizează prin program sau bandă pilot.

Pentru imprimare se folosește un tambur pe al cărei suprafață sînt dispuse elicoidal 56 șiruri de cîte 132 de caractere. Acest tambur se rotește continuu. De asemenea, pentru realizarea imprimării mai există 132 ciocănele. Rîndul de imprimat este conținut într-o memorie tampon de 132 de octeți. Pe baza informațiilor conținute în fiecare octet al

Scanned with OKEN Scanner

acestui tampon se dă o comandă în care un ciocănel lovește hîrtia în dreptul caracterului de pe tambur corespunzător celui din memoria tampon. Astfel fiecare din cele 132 de caractere este imprimat. La o rotire a tamburului este imprimat un rînd pe imprimantă.

4) *Unitatea de bandă magnetică* permite lucrul cu benzi magnetice cu 7 sau 9 piste, cele mai utilizate fiind cele cu 9 piste. Banda magnetică este acoperită cu un strat de oxid magnetic care permite înscrierea de informații. Lungimea maximă a unei benzi magnetice poate fi de 732 m. Pot fi folosite benzi de această lungime sau benzi de lungimi mai mici. Densitatea de înscriere pe bandă poate fi de 800 octeți pe inch sau 1600 octeți pe inch ceea ce revine la 31 respectiv 63 octeți pe mm. Rezultă că pe o bandă pot fi scrise maximum 22 respectiv 44 milioane octeți.

Banda magnetică se montează pe o unitate de bandă magnetică prevăzută cu un derulor care permite trecerea benzii prin fața unor capete de citire / scriere. Înregistrarea se face prin magnetizarea suportului.

În cazul benzilor cu 9 piste, aceste piste corespund la 8 biți plus un bit de paritate care permite verificarea corectitudinii transmiterii informației. Viteza de defilare a benzii prin fața celor 9 capete de citire/scriere (cîte unu pentru fiecare pistă) este de aproximativ 190 cm/s. Derulorul de bandă permite operațiile de scriere/citire, ștergere și rebobinare. Viteza de transfer este de aproximativ 60 kocteți/s.

După cum vom vedea ulterior, informația înscrisă pe bandă este împărțită în blocuri. Acestea sînt despărțite între ele prin spații de aproximativ 15 mm. Deci cantitatea de informație care poate fi înscrisă pe bandă depinde de lungimea blocurilor. Cu cît lungimea blocurilor va fi mai mare, cu atît cantitatea de informație pe care o poate conține banda este mai mare. Pe de altă parte însă, blocuri de lungime prea mare fac ca aria din memoria internă folosită de utilizator pentru programul și datele cu care lucrează să scadă, deoarece transferul de informații între periferic și memoria internă se face prin intermediul unei *zone tampon* din memo-

Scanned with OKEN Scanner

ria internă, zonă a cărei lungime este lungimea maximă a unui bloc.

Principalul inconvenient al benzilor magnetice este imposibilitatea accesului direct la informație. Aceasta face ca ele să fie folosite în problemele care cer parcurgerea în ordine a informațiilor de pe bandă.

5) *Unitatea de discuri DIAM* reprezintă suportul pentru pachetele de discuri DIAM. Un astfel de pachet de discuri conține 6 discuri montate pe un ax; sînt disponibile 10 fețe deoarece fața superioară și cea inferioară a pachetului de discuri nu sînt folosite pentru stocarea informației.

Unitatea de discuri DIAM permite montarea și demontarea unui pachet de discuri (de aceea folosim termenul de discuri mobile).

O față a unui disc conține 200 de piste concentrice plus 6 piste de rezervă, care sînt folosite atunci cînd una dintre cele 200 de piste este defectă. Pistele sînt numerotate de la 0 la 199. Cele 10 piste cu același număr din pachet formează un cilindru. Deci pachetul este constituit din 200 de cilindri, numerotați de la 0 la 199. Fiecare pistă este împărțită în 12 sectoare, numerotate de la 0 la 11.

Un sector conține 256 octeți de informație. Rezultă că o pistă are o capacitate de 3 kocteți, deci un pachet de discuri DIAM are o capacitate de 6000 kocteți.

Unitatea de discuri cuprinde un sistem de 5 brațe prevăzută la extremități cu cîte două capete de citire/scriere. Prin mișcarea acestui sistem de brațe poate fi făcută alegerea unui cilindru. Pista de cilindru este aleasă prin activarea unuia sau altuia dintre capete; locul de pe pistă poate fi ales prin rotația pachetului de discuri în jurul axului (se execută 2400 rotații/m).

Datorită caracteristicilor acestui dispozitiv fizic, informația este înregistrată pe cilindri pentru a evita pe cît posibil mișcările mecanice, deci pentru a mări viteza de citire/scriere.

Viteza de transfer a informației este de 156 kocteți pe secundă; timpul mediu de acces la informația de pe disc este de 85 ms. Timpul foarte redus de acces la informație se datorează posibilității de *acces direct* (spre deosebire de

Scanned with OKEN Scanner

benzile magnetice unde accesul se poate face numai secvențial).

6) *Discuri DIMAS.* Pachetele de discuri DIMAS sînt asemănătoare din punct de vedere al concepției cu pachetele DIAM.

Fiecare pachet de discuri DIMAS MD25 are 203 cilindri (din care 3 de rezervă), fiecare cilindru fiind compus din 20 de piste.

Fiecare pistă este compusă din 6 sectoare a cîte 1024 octeți. Capacitatea unui asemenea pachet de discuri este 24576 kocteți, timpul de acces fiind aproximativ același.

Fiecare pachet de discuri DIMAS MD50 are 400 cilindri plus 6 cilindri de rezervă. Capacitatea lui este de 49 152 kocteți.

7) *Consola* este o mașină de scris care permite operatorului să primească și să transmită mesaje sistemului de calcul.

Dacă facem o comparație între vitezele de lucru, putem aprecia că unitatea de comandă lucrează extrem de repede, iar perifericele lucrează repede.

Perifericele se pot împărți în: rapide (benzile și discurile magnetice) și periferice lente (celelalte periferice).

CONFIGURAȚIA UNUI SISTEM DE CALCUL

Prin configurația de calcul înțelegem totalitatea dispozitivelor prezente în sala calculatorului.

Sistemul FELIX are o natură *modulară*, înțelegînd prin aceasta că o configurație dată poate fi modificată prin adăugirea de noi dispozitive.

Dăm mai jos structura unui sistem de calcul (aceasta este configurația sistemului de calcul al Universității București):

1 unitate centrală
1 consolă
1 cititor de cartele
1 imprimantă

Scanned with OKEN Scanner

1 cititor de bandă perforată
1 perforator de cartele
4 unități de discuri DIAM
4 unități de benzi magnetice
Ele sînt legate conform schemei din figura 4.

DESFĂȘURAREA OPERAȚIILOR DE INTRARE/IEȘIRE

Unitățile de schimburi multiple funcționează după un program propriu, format din *comenzi*. Acest program de lucru se află în memoria internă.

Dacă trebuie executată o operație de intrare/ieșire, este alcătuit programul de lucru al unității de schimburi multiple, introducînd în acest program informații cum ar fi tipul organului periferic, numărul său (dacă există mai

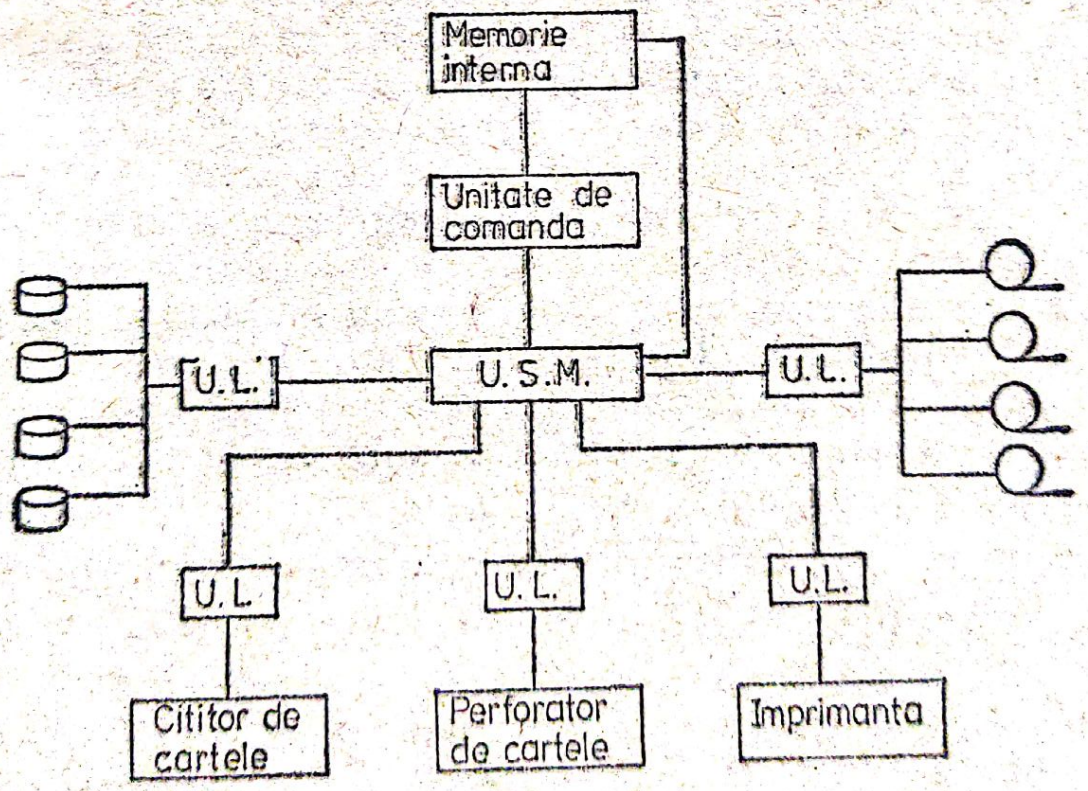


Fig. 4 Exemplu de configurație a unui sistem de calcul

Scanned with OKEN Scanner

multe periferice de același tip), dacă este vorba de o operație de intrare sau de ieșire, lungimea înregistrărilor care trebuie transmise etc.

Unitatea de comandă execută o instrucțiune prin care informează o anumită unitate de schimburi multiple că trebuie să efectueze un transfer de date. Unitatea de schimburi multiple, dacă este liberă, răspunde printr-un semnal. Din momentul primirii acestui semnal, unitatea de comandă nu mai este ocupată de operația respectivă și poate trece la executarea altor instrucțiuni din program, dacă acestea nu depind de transferul de informații inițiat. Totuși, unitatea de comandă poate să intervină pe parcursul transferului de informații pentru a cunoaște stadiul lui sau starea perifericului afectat.

Dacă perifericul este disponibil, el se conectează la unitatea de schimburi multiple prin intermediul unei unități de legătură și transferul de date poate începe. Din informația obținută de la unitatea de comandă, unitatea de schimburi află unde se află în memorie programul său de lucru. Ea citește fiecare comandă, verifică validitatea ei, o decodifică în ordine care sînt executate de unitățile de legătură. Comenzile conțin informații asupra tipului operației, adresei din memorie de unde trebuie luate datele (în cazul operațiilor de ieșire) sau unde trebuie introduse datele (în cazul operațiilor de intrare), locului de pe mediul extern de unde se citesc sau unde se înscriu datele (de exemplu în cazul discurilor numărul cilindrului, al pistei și al sectorului) etc.

În momentul în care operația de intrare/ieșire este încheiată normal sau anormal (datorită unui incident), unitatea de schimburi comunică acest lucru unității de comandă printr-un semnal corespunzător fiecărei situații.

Scanned with OKEN Scanner

Capitolul 2

SISTEMUL DE OPERARE SIRIS

GENERALITĂȚI

Un *sistem de operare* este o colecție de programe care permite rularea programelor utilizatorului. Aceste programe se află pe un anumit disc numit *discul sistem*. Sistemul de operare este livrat odată cu calculatorul. În el pot fi făcute ulterior modificări.

Sistemul de operare trebuie să corespundă caracteristicilor fizice ale calculatorului. Pe de altă parte un calculator electronic „brut" poate fi făcut să lucreze folosind diferite sisteme de operare.

Programele cuprinse în sistemul de operare sînt continuu modificate de firma constructoare sau de către programatorii de sistem din fiecare centru de calcul, modificările vizînd optimizarea programelor componente cu scopul reducerii timpului de rulare a programelor utilizatorilor, îmbunătățirea mesajelor furnizate de sistem referitoare la erorile detectate în programele utilizatorilor, simplificarea sarcinii operatorilor prin creșterea gradului de automatizare al executării lucrărilor, ușurarea scrierii programelor, îmbogățirea facilităților de lucru oferite utilizatorilor, etc. Este indicat ca utilizatorii să caute să fie la curent cu modificările din sistemul de operare pentru a putea folosi noile facilități. În general, modificările care se fac păstrează și vechile facilități, urmărind numai adăugarea unora noi și nu și suprimarea celor vechi, chiar dacă acestea sînt înlocuite cu unele mai performante; din păcate, această regulă nu este totdeauna respectată, ceea ce face cu atît mai necesară cunoașterea de către utilizatori a ultimei variante a sistemului de operare și a modificărilor aduse acestuia.

Scanned with OKEN Scanner

Principalele funcții ale unui sistem de operare sînt următoarele:

1) *Gestionarea resurselor unității de comandă.* Prin aceasta se urmărește asigurarea funcționării continue a unității de comandă, evitînd timpii morți datorați de exemplu așteptării realizării transferului de date între memoria centrală și periferice. Pentru aceasta este folosită *multiprogramarea.*

În esență, multiprogramarea constă în posibilitatea existenței în memoria internă a mai multe programe care trebuie executate, astfel încît dacă la un anumit moment de timp unul dintre ele se află în stare de așteptare, să se treacă la executarea unui alt program, ordinea de executare fiind stabilită conform unor priorități.

2) *Gestionarea resurselor dispozitivelor periferice.* Această funcție constă în capacitatea sistemului de a primi de la un program cererile pentru operațiile de intrare/ieșire, de a urmări executarea de către periferice a acestor operații, de a asigura un sistem de așteptare a cererilor în cazul în care perifericul care trebuie să execute operația este ocupat; astfel, în cazul în care folosim multiprogramarea, sistemul trebuie să fie capabil să reia executarea unui program aflat în stare de așteptare în momentul în care cauzele care au generat această stare au fost rezolvate.

3) *Automatizarea efectuării prelucrărilor.* Prin aceasta se caută să se reducă la maximum și să se simplifice munca operatorului. Astfel în timpul rulării unui program și pentru a asigura rularea programelor unul după altul, operatorul nu mai trebuie să intervină. Aceasta este posibil prin existența unui limbaj de comandă cu ajutorul căruia programatorul poate să-și descrie lucrarea și prin asigurarea unei rulări în flux continuu (începerea rulării unui nou program nu necesită intervenția operatorului). Folosind limbajul de comandă, utilizatorul are posibilitatea să alcătuiască lucrări complexe în care să efectueze mai multe prelucrări. Trecerea de la o prelucrare la alta este făcută automat de către sistemul de operare, ca dealtfel și trecerea de la o lucrare la alta (înlănțuirea lucrărilor). Fiecare sistem de operare trebuie să-și pla-

Scanned with OKEN Scanner

nifice lucrul. Această planificare se face folosiiid o *unitate de planificare* care, din punctul de vedere al sistemului de operare, constituie o prelucrare completă (început, prelucrare, sfîrșit). Aceste unități sînt privite ca independente între ele de cele mai multe sisteme de operare.

4) *Organizarea și accesul la datele de prelucrat.* Această funcție are ca scop de a oferi posibilitatea lucrului cu un volum mare de date prin crearea și exploatarea de fișiere care cuprind aceste date. Fișierele sînt organizate în majoritatea cazurilor pe benzi și discuri magnetice. Sistemul trebuie să fie capabil de a prelucra datele din fișiere conform indicațiilor date de utilizator prin intermediul limbajului de care am vorbit mai sus.

5) *Simplificarea activității de programare.* Aceasta se realizează prin capacitatea sistemului de a rula programe scrise în limbaje de programare evoluate, prin asigurarea posibilităților de „translatare" a unor astfel de programe în codul intern al calculatorului, prin detectarea și afișarea unor mesaje despre erorile sintactice și logice din programul rulat.

Sistemul de operare mai contribuie la simplificarea activității de programare prin asigurarea posibilității de a folosi bibliotecile de programe (sursă, binar translatabile sau imagine memorie translatabilă).

6) *Teleprelucrarea.* Sistemul de operare poate executa lucrări care sînt introduse de la distanță prin dispozitive periferice legate la calculator prin linii de transmisie. Rezultatele lucrării sînt transmise de asemenea la distanță utilizatorului. În acest mod se poate asigura accesul rapid la resursele sistemului de calcul al unor programe aflate în alt loc decît acela unde e instalat sistemul de calcul.

STRUCTURA SISTEMULUI DE OPERARE SIRIS

În continuare nu vom vorbi despre sisteme de operare în general, ci vom trece la descrierea sistemului de operare SIRIS al calculatorului FELIX C-256.

Scanned with OKEN Scanner

Sistemul de operare SIRIS a cunoscut o evoluție rapidă oferind cu fiecare nouă versiune noi facilități utilizatorului.

În momentul actual, în majoritatea centrelor de calcul există în lucru versiunile SIRIS-2 versiunea 14.2 și SIRIS-3 versiunea 15. SIRIS-3 urmează în mod logic sistemului SIRIS-2, îmbunătățindu-l și aducîndu-i modificări.

Descrierea pe care o vom da sistemului de operare SIRIS va fi aplicabilă ambelor variante. Acolo unde este cazul, vom face precizările cuvenite pentru a marca diferențele existente.

Programele cuprinse în sistemul de operare SIRIS sînt grupate în subsisteme, fiecare dintre acestea fiind la rîndul lui format din subsisteme, etc.

Sistemul de operare SIRIS are următoarea structură:

I. *Sistemul de exploatare*

1. Monitorul
   a) Supervizorul
   b) Monitorul central
   c) Monitorul de înlănțuiri
2. Editorul de legături
3. Bibiotecarul
4. Generatorul de sistem

II. *Sistemul de programare*

1. Compilatoarele și asamblorul
2. Macroprocesoarele

III. *Sistemul de gestiune al fișierelor (SGF)*

LUCRARE ȘI FAZĂ

O lucrare (Job) reprezintă un ansamblu de activități realizate de sistemul de operare; o lucrare este o succesiune de instrucțiuni ale limbajului prin care utilizatorul se adresează sistemului împreună cu datele pe care le prelucrează diferitele programe ale sistemului (de exemplu programul scris de utilizator într-un limbaj de programare evoluat sau

Scanned with OKEN Scanner

datele de intrare pentru acest program), instrucțiuni cuprinse între o cartelă JOB și prima cartelă EOJ care îi urmează.

Lucrarea constituie pentru sistemul de operare SIRIS unitatea de planificare.

O lucrare poate cuprinde una sau mai multe faze. Prin fază vom înțelege una din prelucrările următoare:

— compilarea unei unități de program
— editarea legăturilor
— încărcarea și execuția unui program IMT.

Vom descrie pe scurt principalele faze ale unei lucrări:

1) *Compilarea.* Aceasta constă în traducerea unităților programului sursă scrise într-un limbaj de programare evoluat. Ca rezultat se obține reprezentarea binară a instrucțiunilor în limbaj mașină. Adresele alocate diferitelor date sau instrucțiuni din fiecare unitate de program sînt stabilite relativ la unitatea de program respectivă și nu reprezintă adrese în memoria internă a calculatorului; aceasta se datorește faptului că rezultatul compilării nu este direct executabil, deci nu poate fi încărcat direct în memoria internă pentru a fi executat (reamintim faptul că orice program, fie al utilizatorului, fie al sistemului de operare trebuie să fie încărcat în memoria internă pentru a putea fi executat).

Fiecare unitate de program este compilată independent, ceea ce face ca legăturile dintre unitățile de program să nu fie satisfăcute în această fază. De asemenea, nu se stabilesc legături cu subrutinele standard (radical, logaritm, etc.) și cu subrutinele sistemului necesare lucrului cu fișiere.

Vom numi *modul obiect* rezultatul compilării unei unități de program. Spunem că modul obiect este în *format B.T.* (binar translatabil) deoarece modulul obiect, după o fază prealabilă, este încărcat în memorie începînd de la o anumită adresă și deci adresele relative stabilite la compilare vor fi translatate, corespunzător adresei de încărcare.

Compilarea se execută de către un program translator *(compilator)* al sistemului, corespunzător limbajului în care a fost scris programul sursă. Compilatorul pentru limbajul ASSIRIS se mai numește *asamblor*.

2) *Editarea legăturilor.* În această fază se obține un program executabil (numit *program obiect*)din mai multe module obiect. Acest lucru este realizat de un alt program al sistemului, numit *editorul de legături*. Editorul de legături corelează modulele obiect între ele prin satisfacerea conexiunilor încorporează subrutinele standard care sînt folosite și subrutinele sistemului necesare creării și exploatării fișierelor. Rezultatul este un program direct executabil de către calculator, deci un program exact în forma în care va apare în memoria internă în momentul execuției. De aceea spunem că programul obiect este în *format I.M.T.* (imagine translatabilă a memoriei).

3) *Executarea.* Această fază constă în stabilirea unei adrese începînd de la care să fie încărcat un program obiect, încărcarea lui la acea adresă și executarea lui.

Menționăm aici că sistemul SIRIS permite segmentarea unui program. Prin aceasta înțelegem în primul rînd descompunerea programului în părți numite *segmente*. La fiecare moment de timp în memorie va fi prezent numai un anumit număr de segmente (ideea segmentării fiind aceea de a permite rularea unor programe, care, datorită numărului mare de instrucțiuni sau volumului mare de date pe care le prelucrează, nu încap în totalitatea lor în memoria internă). Segmentele vor fi încărcate în memorie numai atunci cînd este necesar, eventual peste alte segmente existente în acel moment în memorie, dar a căror prezență nu mai este necesară.

Stabilirea adreselor de încărcare este făcută de către editorul de legături pe baza unei structuri de acoperire a segmentelor furnizată de utilizator. Încărcarea segmentelor se face în timpul execuției, de către monitor.

Trecerea unui program prin etapele descrise mai sus asigură o mare flexibilitate. Programele sursă, modulele obiect și programele obiect pot fi catalogate în biblioteci sau scoase pe cartele perforate, putînd fi ulterior folosite. Un modul obiect poate fi necesar mai multor utilizatori și atunci pre-

Scanned with OKEN Scanner

zența lui într-o bibliotecă îi scutește de munca de a-l rescrie; în plus, existența unui modul obiect (folosit separat) într-o bibliotecă sau pe cartele evită etapa de compilare, reducînd astfel timpul de lucru. Catalogarea unui program obiect oferă posibilitatea ca el să fie încărcat și executat ori de cîte ori este nevoie (cu date de intrare diferite) fără a mai trece prin etapele de compilare și editare de legături.

TIPUL LUCRĂRILOR ȘI GESTIONAREA MEMORIEI

După cum am amintit mai înainte, sistemul SIRIS permite prezența în memoria internă a mai multor programe, executarea lor făcîndu-se conform unor priorități. Pentru a fi posibil acest lucru, memoria este împărțită în zone numite partiții. Fiecărei partiții i se asociază o cheie de protecție cu ajutorul căreia sistemul păstrează împărțirea memoriei și împiedică accesul la conținutul unei partiții din interiorul alteia, preîntîmpinînd astfel distrugerea din greșeală a unor informații memorate. Împărțirea în zone și dimensiunile acestora se stabilesc la inițializarea sistemului, dar pot fi modificate de către operator în timpul exploatării (numim acest mod împărțirea *statică* a memoriei, pentru a o deosebi de împărțirea *dinamică* în care divizarea memoriei se face în funcție de programele care se încarcă; sistemul SIRIS permite numai împărțirea statică a memoriei). De asemenea, la generare, fiecărei partiții i se asociază o prioritate, care reprezintă și prioritatea programului încărcat în partiția respectivă (într-o partiție poate fi încărcat și executat un singur program).

Memoria internă este împărțită în pagini, o *pagină* cuprinzînd 2048 octeți. O partiție este constituită dintr-un număr întreg de pagini. Există următoarele tipuri de partiții posibile:

1) *Partiția monitor.* Aceasta este prezentă totdeauna și este constituită din acele programe ale monitorului care sînt folosite foarte des și de aceea trebuie să se afle în permanență în memorie (încărcarea lor de fiecare dată cînd

este necesar ar mări timpul de lucru și nici nu ar constitui o economie deosebită de memorie, în general fiind nevoie de toate aceste programe ale sistemului). Mărimea ei este fixată atunci cînd este generat sistemul de operare.

2) *Partiție serie* în care pot fi executate lucrări care pot avea mai multe faze. Aceste faze pot fi compilare, editare de legături sau execuție.

3) *Partiție paralelă* în care se pot executa lucrări ce constau din mai multe faze. Aceste faze nu pot fi compilări sau editări.

4) *Partiție paralelă* în care se pot executa numai lucrări ce constau dintr-o singură fază de execuție.

5) *Partiție de timp real.* Lucrările tratate în această partiție se numesc *lucrări în timp real* și reprezintă lucrări a căror rezolvare trebuie să se facă în cel mai scurt timp posibil (luarea fără întîrziere de decizii asupra modului de continuare a unui proces pe baza datelor obținute pînă la acel moment de timp).

Într-o partiție serie lucrările sînt executate în serie (înlănțuite). De acest lucru se ocupă monitorul de înlănțuiri.

O lucrare în paralel poate fi adusă într-o partiție paralelă prin intervenția operatorului, cu ajutorul unei cartele de comandă sau prin intermediul unei lucrări în curs de execuție. Încărcarea se face plecînd de la biblioteca utilizator standard (despre care se va vorbi tot în acest capitol) sau de la o bibliotecă oarecare.

Stabilirea mărimii unei partiții cu înlănțuire la generarea sistemului de operare sau a spațiului alocat la inițializarea sistemului de operare este în funcție de ce lucrări vor fi rulate în ea și de tipul discului sistem (DIMAS sau DIAM).

Partițiile destinate unor lucrări ce constau dintr-o singură fază de execuție pot avea o mărime mică în funcție de mărimea fazei (4 K, 6 K...)

În tabelul următor dăm mărimea minimă a partiției în diferite cazuri:

Scanned with OKEN Scanner

Sistemul de operare	Tipul partiției	Disc sistem	
		DIAM	DIMAS
SIRIS 3	serie	26 K	36 K
	paralelă cu înlănțuire	24 K	32 K

Observații:

1. În documentația referitoare la sistemul SIRIS-3, în locul denumirii de partiție este utilizată aceea de *regiune*. Tipul unei regiuni depinde de ce lucrări vor putea fi executate în aceasta.

2. Pentru sistemul de operare SIRIS-2 poate exista o singură partiție serie.

3. În sistemul de operare SIRIS-3, în cazul cînd memoria e suficientă, pot exista mai multe regiuni în care să se execute lucrări cu toate cele trei faze.

4. În sistemul SIRIS actual numărul maxim al partițiilor poate fi 12.

GESTIUNEA LUCRĂRILOR

Orice lucrare care va fi executată va utiliza resursele calculatorului. Din acest punct de vedere putem deosebi lucrările între ele prin:

— mărimea memoriei necesare pentru execuție

— ce dispozitive periferice utilizează

— prioritatea stabilită în funcție de timpul în care se dorește obținerea rezultatelor.

După aceste caracteristici, lucrările pot fi împărțite în mai multe clase, în fiecare clasă fiind lucrări cu caracteristici asemănătoare (compilatoare utilizate, necesar de memorie).

Scanned with OKEN Scanner

Această clasificare are importanță pentru utilizarea eficientă a sistemului de calcul.

Împărțirea efectivă a lucrărilor în clase se poate face de către personalul calculatorului sau de către sistemul de operare însuși, pe baza unor informații furnizate de utilizator (numărul clasei din care face parte, prioritatea, caracteristicile fișierului de ieșire etc.).

În sistemul de operare SIRIS-2 această operație nu e realizată de către sistemul de operare, spre deosebire de SIRIS-3.

Modul obișnuit de gestionare a lucrărilor este cel din figura 6. După citirea lucrărilor și execuția lor rezultatele vor fi afișate pe un dispozitiv de ieșire.

Pentru aceasta, în sistemul de operare sînt folosite mai multe fișiere, a căror descriere o vom da în continuare.

FIȘIERELE SISTEMULUI

1) *Fișierul de lucrări* constituie sursa de alimentare a unei partiții serie. Informațiile care pot apare în fișierul de lucrări sînt următoarele:

— cartele de comandă prin care programatorul indică monitorului operațiile care trebuie efectuate de către sistem;

— programe scrise într-un limbaj de programare (care urmează să fie compilate, deci constituie datele pentru compilatoare sau asamblor);

— datele de intrare care trebuie citite și prelucrate de programe în momentul execuției.

Fișierul de lucrări conține un șir de lucrări care sînt prelucrate secvențial, această sarcină fiind dusă la îndeplinire de către monitorul de înlănțuiri; astfel este ușurată foarte mult munca operatorului.

Fișierul de lucrări este creat pe unul din suporturile: cartele (deci citit de la cititorul de cartele), bandă sau disc magnetic.

Ca orice fișier, el este identificat printr-un *idex* (identificator de exploatare) pe care îl folosim cînd vrem să

facem referiri la el. În cazul fișierului de lucrări idexul este *1.

Conținutul unui fișier de lucrări poartă numele de *lot de lucrări* sau *lanț de lucrări*.

2) *Fișierul de ieșire prin imprimare* are idexul *2. Aici apar informațiile date de sistem utilizatorului asupra compilării, editării legăturilor, erorilor din program, rezultatele executării programelor, etc. De obicei, suportul acestui fișier este imprimanta, dar poate fi și o bandă magnetică sau un disc magnetic.

3) *Fișierul de ieșire prin perforare* are idexul *3. El este format din imagini ale cartelelor perforate. De obicei, acest fișier are ca suport cartele, dar poate avea ca suport și benzi sau discuri magnetice. El este folosit de exemplu în următoarele situații:

— scoaterea pe cartele a rezultatelor unui program, rezultate care să constituie datele de intrare pentru un alt program (în acest mod se evită erorile de perforare și de conversiune de la o bază la alta, care apar la listarea rezultatelor la imprimantă și perforarea lor ulterioară pe cartele);

— obținerea pe cartele a modulelor obiect sau a programului obiect.

4) *Fișierul FILEDIT* este fișierul de intrare al editorului de legături. Aici trebuie aduse toate modulele obiect care urmează să fie editate. Acest fișier se găsește întotdeauna pe disc în zona de manevră standard Z%MANS și este creat de monitorul de înlănțuiri. Un modul obiect poate fi adus în fișierul FILEDIT în urma compilării, prin copiere din fișierul de lucrări sau dintr-o bibliotecă. Acestui fișier nu i se atașează un idex deoarece nu se pot face referiri la el din cadrul programului.

5) *Fișierul LOADGO* este fișierul de ieșire al editorului de legături și are idexul *4. El se află pe discul sistem. Aici va apare programul obiect în format IMT obținut prin editare. De aici, programul obiect poate fi încărcat în memorie, catalogat într-o bibliotecă sau scos pe cartele.

Acest fișier este situat de asemenea pe disc. El nu e distrus decît la următoarea editare de legături. Deci este posibilă

Scanned with OKEN Scanner

executarea aceluiași program de mai multe ori (eventual cu date diferite) folosind fișierul LOADGO fără să fie necesară catalogarea programului într-o bibliotecă IMT.

Suportul standard pentru fișierele de intrare și de ieșire (hîrtie imprimantă și cartele) este fixat la inițializarea sistemului. Pentru FILEDIT și LOADGO suportul fiind standard, se pot modifica doar dimensiunile și unitatea pe care se găsesc aceste fișiere.

Modul de gestiune al lucrărilor prezentate mai sus este propriu sistemului de operare SIRIS-2. Sistemul de operare SIRIS-3, pe lîngă această posibilitate de gestiune a lucărilor mai poate folosi și alta cu performanțe superioare — SYMBIONT și TELESYMBIONT.

Fiecărei lucrări i se atașează de către utilizator o serie de informații referitoare la caracteristicile ei (clasa din care face parte, ce caracteristici va avea dispozitivul de extragere a rezultatelor etc.).

Pe măsură ce lucrările sosesc în sistem, ele sînt prelucrate de SYMBIONT (cele locale) și TELESYMB (cele de la distanță). Pe baza informațiilor furnizate, fiecare lucrare va fi pusă într-o clasă de lucrări. Lucrările din fiecare clasă se vor executa într-o regiune care i-a fost atribuită acesteia în ordinea priorității lor. Deci lucrările din fiecare clasă vor intra în execuție în ordinea dată de prioritățile lor.

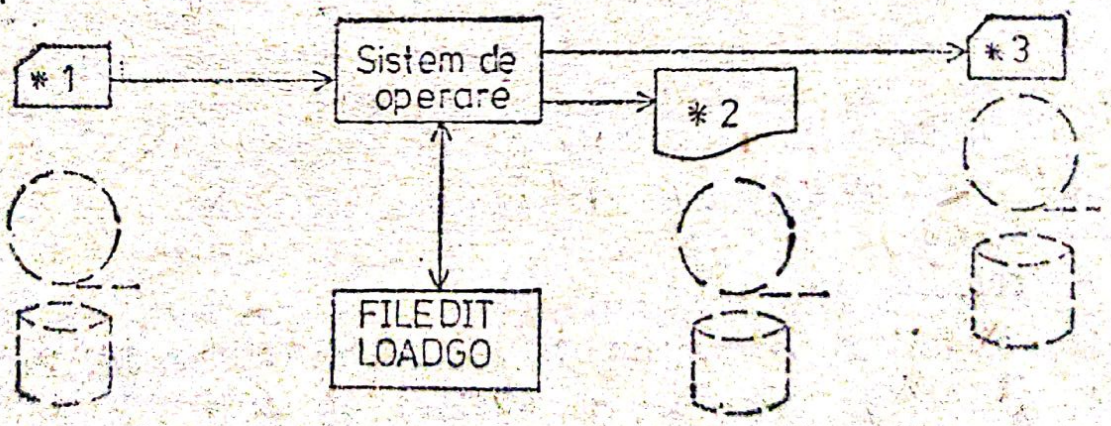


Fig.6. Fișierele sistemului

După executarea acestor lucrări, rezultatul este pus într-un fișier de ieșire.

Conținutul unui fișier de ieșire va fi afișat pe un anumit dispozitiv periferic care îi este atribuit la pornirea lucrului în sistemul de operare.

În funcție de parametrul dat de utilizator care exprimă caracteristicile pe care trebuie să le aibă rezultatul lucrării sale (hîrtie cu dublu exemplar, hîrtie simplă, hîrtie specială preimprimată în 4 exemplare etc.), rezultatul execuției lucrării va fi pus în acel fișier în care lucrările dintr-o anumită categorie așteaptă ca perifericul care are caracteristicile dorite de utilizator să fie liber. Lucrările din fiecare fișier sînt imprimate în momentul cînd dispozitivul periferic este liber (a terminat o extragere a rezultatelor dintr-un alt fișier care i-a fost asociat). Modul de lucru e reprezentat în figura 7.

BIBLIOTECILE SISTEMULUI

Prin *bibliotecă* înțelegem o colecție de programe, segmente sau module, în diferite formate, organizată pe un suport magnetic.

Vom face distincție între bibliotecile de programe pe care utilizatorul are posibilitatea să le creeze și să le exploateze (numite *biblioteci utilizator)* și *bibliotecile sistemului* care sînt create odată cu sistemul. Ambele tipuri de biblioteci pot fi create și exploatate cu ajutorul programului bibliotecar (dar bibliotecile sistemului pot fi utilizate direct și de către alte componente ale sistemului).

Bibliotecile sistemului se află pe discul sistem. Ele sînt următoarele:

1) *Biblioteca utilizator standard* (B.U.S.) conține programe în format I.M.T. Din ea fac parte programele sistemului SIRIS. În această bibliotecă pot fi catalogate și programe obiect ale utilizatorului, catalogarea putîndu-se face cu ajutorul bibliotecarului sau cu ajutorul editorului de legături.

Numele bibliotecii B.U.S. este Z%BUS.

Programele obiect din această bibliotecă pot fi lansate în regiuni fără înlănțuire sau în timp real cu comanda EXECUTE sau prin intervenția operatorului; în regiunea de exploatare ele pot fi lansate prin RUN sau SYSRUN.

2) *Biblioteca standard de subprograme* (B.S.S.) conține subprograme în format B.T. Această bibliotecă este exploa-

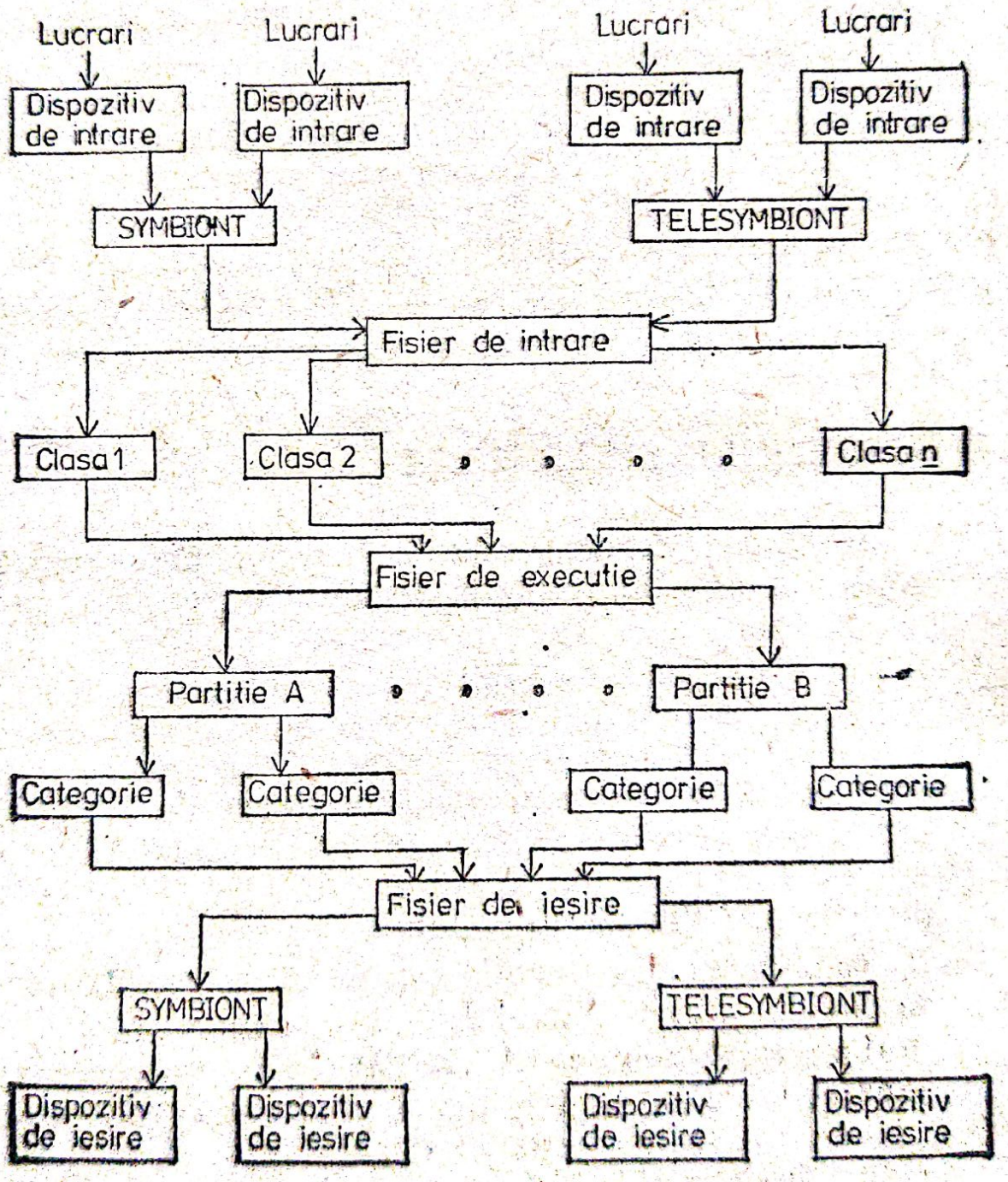


Fig 7 Gestiunea lucrarilor in SIRIS 3 (SYMBIONT si TELESYMBIONT)

Scanned with OKEN Scanner

tată în general de editorul de legături. Printre altele, ea conține subprograme de conversiune FORTRAN, subprograme științifice FORTRAN, module ale sistemului de gestiune a fișierelor, etc.

Accesul editorului de legături la această bibliotecă este subînțeles, dar el poate fi interzis prin folosirea opțiunii NSLB în cartela LINK.

3) *Biblioteca procedurilor catalogate* este destinată pentru păstrarea procedurilor catalogate, cu ajutorul bibliotecarului.

BIBLIOTECARUL

Programul bibliotecar este acea parte a sistemului de operare care permite (prin programele care îl alcătuiesc) crearea și exploatarea bibliotecilor. O bibliotecă este o colecție de programe în unul dintre formatele cunoscute. Bibliotecarul poate realiza catalogarea unui program într-o bibliotecă, ștergerea sau modificarea programului catalogat, compararea a două programe dintr-o bibliotecă etc.

Funcțiile sale și modul în care se comandă executarea lor vor fi descrise într-un capitol separat.

MONITORUL

După cum am văzut, monitorul este compus din trei părți:
— supervizor
— monitor central
— monitor de înlănțuiri.

1) *Supervizorul*

Este compus din trei module care-i permit să:
— trateze întreruperile și devierile
— lanseze un alt program care așteaptă, în cazul apariției unei întreruperi sau abateri
— conducă operațiile de intrare/ieșire fizice.

În cazul întreruperilor, supervizorul prin modulul specializat în acest scop va salva mai întîi o serie de informații ne-

Scanned with OKEN Scanner

cesare pentru reluarea lucrării, va identifica întreruperea sau devierea respectivă lansînd rutina care o va prelucra.

A doua funcție, care e realizată de modulul supervizorului numit *distribuitor* are sens cînd se lucrează în multiprogramare. Dacă în timp ce un program este rulat într-o anumită partiție apare o întrerupere sau abatere, acest modul al supervizorului va analiza ce lucrări așteaptă pentru a fi executate în celelalte partiții, alege pe cea cu prioritatea cea mai mare și o lansează. Acest mod de lucru permite o mai bună utilizare a calculatorului, deoarece în timp ce un program așteaptă (de exemplu pentru îndeplinirea unei operații de intrare/ieșire), un alt program este executat.

Al treilea modul realizează gestiunea intrărilor/ieșirilor. Cererile de efectuare a unui transfer de date sînt păstrate în fire de așteptare proprii fiecărui periferic. În momentul cînd apare o întrerupere de sfîrșit de intrare/ieșire, supervizorul ia act că un periferic a devenit liber și că poate să lanseze următoarea operație de intrare/ieșire relativă la acest periferic. Odată lansată, operația de intrare/ieșire se poate desfășura concomitent cu alte operații efectuate de către unitatea de comandă. Bine utilizată, această posibilitate duce la un timp mai mic de execuție a unei anumite lucrări.

2) *Monitorul central*

Programele monitorului central se încarcă în partiția monitor. Unele dintre ele apar în permanență în memoria internă (deoarece sînt folosite foarte des), iar celelalte sînt încărcate în memoria internă numai atunci cînd este nevoie de ele (sînt încărcate într-o zonă a partiției monitor numită *zonă de tranziție*).

În momentul cînd un program din partiția serie este terminat, controlul este trecut monitorului central care încarcă monitorul de înlănțuiri în această partiție; monitorul de înlănțuiri asigură în continuare înlănțuirea automată a lucrării sau fazei care urmează.

Analog, atunci cînd o lucrare în paralel este terminată, monitorul central preia controlul și întreprinde o serie de acțiuni, dintre care amintim eliberarea resurselor (periferi-celor) ocupate de lucrarea respectivă.

În cazul în care folosim segmentarea programelor, monitorul central este acela care încarcă segmentele conform indicațiilor utilizatorului.

Operatorul poate comunica cu sistemul cu ajutorul monitorului. Pentru aceasta, operatorul apasă pe butonul de întrerupere de pe panoul de comandă sau de pe pupitrul de comandă, monitorul listează la consolă un mesaj pentru a indica faptul că a luat cunoștință de întreruperea solicitată; în continuare, operatorul scrie la consolă informațiile pe care dorește să le comunice sistemului, informații a căror validitate este controlată de monitorul central; dacă informațiile sînt valide, monitorul central întreprinde acțiunile necesare pentru executarea lor. În acest mod, operatorul poate de exemplu introduce un program într-o partiție paralelă.

Un alt rol al monitorului central constă în gestiunea ceasurilor sistemului. Acestea sînt în număr de trei și cu ajutorul lor se poate măsura timpul scurs între două evenimente, se poate măsura cît timp este ocupată unitatea centrală, se pot fixa intervale de timp în care trebuie să se producă un anumit eveniment (de exemplu se poate fixa un interval de timp în care, dacă execuția nu s-a încheiat, programul este întrerupt).

3) *Monitorul de înlănțuiri* este acea parte a monitorului care asigură înlănțuirea automată a lucrărilor și fazelor într-o regiune.

Inițial, monitorul de înlănțuiri este încărcat în memorie de programul de inițializare al sistemului; apoi el este încărcat la sfîrșitul fiecărei faze de către monitorul central. Încărcarea se face într-o partiție serie.

Legătura între programator și monitorul de înlănțuiri se realizează prin intermediul unor *cartele de comandă* (care conțin instrucțiuni în limbajul monitorului de înlănțuiri). Aceste cartele de comandă trebuie să apară de asemenea în fișierul de lucrări. Monitorul de înlănțuiri citește aceste cartele, verifică sintaxa lor și execută acțiunile pe care ele le cer făcînd apel la programele corespunzătoare ale sistemului. Dacă pe o cartelă apare o informație eronată, monitorul de înlănțuiri

semnalează acest lucru operatorului, tipărește un mesaj și, în funcție de eroare, abandonează cartela respectivă, faza care a conținut-o sau chiar întreaga lucrare.

În cazul segmentării programului, monitorul de înlănțuiri încarcă *segmentul rădăcină* (acel segment al programului, a cărui prezență în memorie este tot timpul necesară, segment care conține întotdeauna programul principal) și transmite comanda acestui segment. În afară de aceste roluri, monitorul de înlănțuiri tratează și procedurile catalogate.

Observație.

În sistemul SIRIS-3, monitorul mai conține și a patra componentă: modulul de gestiune automată a intrărilor și ieșirilor referitoare la lucrările locale sau la distanță (SYMBIONT și TELESYMBIONT).

SISTEMUL DE GESTIUNE AL FIȘIERELOR (S.G.F.)

Această parte a sistemului de operare se ocupă cu tratarea fișierelor. Executarea funcțiilor sale este comandată (în cadrul restrictiv al acestui manual) de instrucțiunile scrise în limbajul monitorului de înlănțuiri.

S.G.F. este format în majoritatea cazurilor din programe în format B.T. catalogate în B.S.S. Ele sînt încorporate la cerere în programul în format I.M.T. de către editorul de legături.

Principalele funcții ale sistemului de gestiune al fișierelor sînt următoarele:

— *asignarea de periferice*, adică stabilirea unei corespondențe: idexul unui fișier ⟷ unul sau mai multe periferice care reprezintă suportul fișierului, corespondență prin care putem face referiri la fișier din cadrul programului;

— inițializarea unui disc sau a unei benzi magnetice (înscrierea pe disc sau bandă a unui identificator care să permită recunoașterea lui);

— *alocarea de spațiu* pentru fișier, adică rezervarea unei zone de pe suport pentru fișierul respectiv;

— *crearea fișierului*;

— *exploatarea fișierului* (citirea și scrierea de informații din și în fișier, actualizarea lui prin înlocuirea unor informații perimate prin altele noi, etc.);

— *eliberarea suportului de fișier* (prin aceasta, zona respectivă de pe bandă sau disc poate fi alocată pentru un alt fișier);

— *protejarea accesului la fișier* conform dorinței programatorului ca fișierul să poată fi exploatat numai prin citire, numai prin scriere sau în ambele moduri;

— *gestiunea zonelor tampon*. Prin *zona tampon* înțelegem o zonă a memoriei interne care servește la transferul de informație între memoria internă și periferice. Prin gestiunea zonelor tampon înțelegem încărcarea (citirea) și vidarea (scrierea) lor, precum și alegerea uneia sau alteia dintre zonele tampon asociate unui fișier (unui fișier îi putem asocia una sau mai multe zone tampon). Această gestiune se face în funcție de organizarea fișierelor, despre care vom vorbi mai tîrziu;

— *detectarea și tratarea erorilor* apărute la exploatarea unui fișier. Erorile sînt semnalate prin tipărirea de mesaje. Sistemul de gestiune al fișierelor, în momentul întîlnirii unei erori, poate trece la corectarea ei, la abandonarea fazei sau a lucrării, în funcție de eroarea respectivă;

— *crearea și exploatarea punctelor de reluare*. Utilizatorul are posibilitatea de a jalona executarea programului său prin crearea de puncte de reluare. În acest mod este posibil ca la întreruperea execuției datorită unor cauze cum ar fi o greșeală a sistemului, un accident sau depășirea timpului permis pentru execuție, să nu mai fie necesară reluarea programului de la început, putîndu-se relua execuția de la un anumit moment, moment pînă la care lucrurile au decurs normal. Prin *punct de reluare* înțelegem o înregistrare a stării programului, a conținutului memoriei interne și a registrelor, a stării periferice și a poziționării lor. Fiecare punct de reluare este identificat printr-un număr de ordine care i se atașează. Reluarea programului se face printr-o instrucțiune a monitorului de înlănțuiri, folosind unul dintre punctele de reluare anterior create, punct de reluare identificat prin numărul care îi

Scanned with OKEN Scanner

este atașat. Reluarea constă în refacerea conținutului memoriei interne și comandarea continuării executării programului.

GENERATORUL DE SISTEM servește la crearea unui nou disc sistem.

În funcție de configurația sistemului de calcul și de tipul lucrărilor care se execută pe un anumit calculator, se poate crea un nou sistem de operare plecînd de la cel furnizat de firma constructoare.

Să presupunem de exemplu că într-un centru de calcul lucrările care trebuie executate nu folosesc niciodată un anumit limbaj de programare sau anumite programe ale sistemului de operare inițial. Atunci putem crea un nou disc sistem care să nu mai cuprindă compilatorul respectiv și nici acele programe ale sistemului care nu sînt folosite. Aceasta poate conduce la o economie de spațiu pe disc, reducerea numărului de programe de pe discul sistem atrăgînd după sine reducerea timpului de găsire a lor.

Scanned with OKEN Scanner

Capitolul III

DESCRIEREA PRINCIPALELOR CARTELE DE COMANDĂ

CONVENȚII DE SCRIERE

Vom începe cu prezentarea convențiilor de scriere care vor fi folosite la prezentarea cartelelor de comandă. Menționăm că aceste convenții sînt valabile și pentru instrucțiunile, ordinele și cartelele parametru care vor fi descrise în această carte.

După cum se știe, parantezele drepte, acoladele și sublinierile nu sînt recunoscute de calculator; de aceea ele nu pot face parte efectiv din text. Apariția lor la prezentarea formei cartelelor de comandă are următoarea semnificație:

— informația inclusă între paranteze drepte este opțională;

— dacă mai multe informații sînt scrise una sub alta și sînt cuprinse între acolade, atunci una și numai una dintre ele trebuie specificată;

— dacă una dintre informațiile cuprinse între acolade este subliniată, atunci nespecificarea niciuneia dintre ele este echivalentă cu specificarea celei subliniate (spunem că aceasta este presupusă prin lipsă).

FORMATUL CARTELELOR DE COMANDĂ

Cartelele de comandă au următoarea formă generală:

ident listă de argumente [*comentariu*]

Punctul apare în coloana 1; *ident* indică tipul operației și este scris începînd din coloana 11; după *ident* trebuie să urmeze un blanc.

listă de argumente este un șir de argumente despărțite între ele prin virgulă; lista de argumente nu poate depăși coloana 71.

Scanned with OKEN Scanner

comentariu este un șir de caractere despărțit de lista de argumente printr-un blanc.

Lista de argumente poate continua și pe cartela sau cartelele următoare. Un asterisc în coloana 72 indică faptul că următoarea cartelă este o cartelă în continuare. O cartelă de continuare conține un punct în coloana 1, următoarele informații continuînd din coloana 20. Un argument trebuie să apară în întregime pe o cartelă.

În mod normal, în lista de argumente nu poate apare un blanc decît dacă acesta face parte dintr-un șir de caractere cuprins între apostrofuri. Dacă însă informația se întinde pe mai multe cartele, virgula care urmează ultimului argument de pe o cartelă poate fi urmat de blancuri (pentru a nu contrazice regula conform căreia un argument trebuie să apară în întregime pe o cartelă); această situație este recunoscută de sistem datorită asteriscului care apare în coloana 72.

Prezentăm mai jos principalele cartele de comandă relative la exploatare.

CARTELA JOB

Această cartelă indică începutul unei noi lucrări în partiția serie. Forma ei este:

JOB *ident*,AN:*n*,PN:*nume*

ident este numele lucrării și este format din 1—8 caractere alfanumerice, dintre care primul este o literă;
n este numărul de cont al programatorului și este format din 4 caractere alfanumerice;
nume este numele programatorului și este format din 1—8 caractere; blancul este exclus.

CARTELA EOJ

Această cartelă indică sfîrșitul lucrării. Forma ei este:

EOJ

CARTELA DE COMENTARII

Această cartelă permite inserarea de comentarii în secvența cartelelor de comandă. Forma ei este următoarea:

- $\left\{\begin{matrix}*\\**\end{matrix}\right\}$ *comentarii*

Cartela este citită de monitorul de înlănțuiri, care o listează pe fișierul de ieșire (și o scrie la consolă dacă tipul operației este **). O astfel de cartelă poate fi folosită de exemplu cînd vrem să transmitem operatorului o anumită informație.

CARTELA LINK

Vom prezenta aici numai o formă simplificată a acestei cartele și anume următoarea:

- LINK *listă de argumente*

Efectul acestei cartele de comandă este editarea legăturilor între diferite module obiect, obținîndu-se un program în format I.M.T. Folosirea argumentului XREF determină listarea unei tabele de referințe încrucișate.

CARTELA RUN

Efectul acestei cartele este încărcarea în memoria internă a unui program în format I.M.T. și executarea lui (este vorba de programul desemnat sau prin cartelă sau printr-o cartelă FETCH sau LINK precedentă). Forma cartelei RUN este următoarea:

- RUN [FN:*ident*] [,NL:*n*] [,TIME:*m*]

$$\left[,\left\{\begin{matrix}UD\\AD\\OD\end{matrix}\right\}:\left\{\begin{matrix}CP\\\\adresa\ 1,\ adresa\ 2\end{matrix}\right\}\right]$$

Scanned with OKEN Scanner

cu observația că primul argument care apare nu poate fi precedat de virgulă.

ident este numele programului a cărui execuție trebuie lansată. Dacă numele programului lipsește, înseamnă că este vorba de programul pentru care tocmai s-a făcut editarea legăturilor (dacă înaintea cartelei RUN apare cartela LINK) sau de programul adus în memorie dintr-o bibliotecă prin cartela FETCH care precede cartela RUN. Facem observația că dacă programul care trebuie încărcat și executat se află în biblioteca B.U.S., apariția numelui său este obligatorie.

n este un număr de 1—6 cifre, mai mic decît 524288, care reprezintă numărul maxim de linii ce pot fi tipărite. Prin lipsă, $n = 1000$.

m este un număr de 3 cifre reprezentînd numărul maxim de minute rezervat pentru desfășurarea fazei. Prin lipsă, $m = 3$.

Celelalte opțiuni cer afișarea memoriei alocate programului, după cum urmează:

UD — în cazul unui sfîrșit normal;

AD — în cazul unui sfîrșit anormal;

OD — în cazul unui sfîrșit anormal, dar numai conform autorizației operatorului, autorizației pe care acesta o dă de la consolă tipărind:

Y pentru DA

N pentru NU

Porțiunea din memorie care urmează a fi listată se specifică prin:

CP — dacă dorim să fie listată întreaga zonă program

adresă 1, adresă 2 dacă dorim să fie listată numai porțiunea cuprinsă între limitele specificate de cele două adrese.

Observație: Dacă programul care trebuie încărcat și executat folosește fișiere, cartelele ASSIGN și LABEL care descriu aceste fișiere trebuie plasate înainte de RUN.

Scanned with OKEN Scanner

CARTELE COMPILE

Această cartelă determină încărcarea în memorie a unui program compilator și executarea lui. În același timp, cartela permite catalogarea într-o bibliotecă B.T. a modulului obiect obținut. Neavînd încă elementele necesare pentru a vorbi de catalogare, vom prezenta cartela COMPILE într-o formă simplificată. Această formă implică faptul că programul sursă trebuie să se afle în fișierul de lucrări.

COMPILE *ident* [,*listă de opțiuni*]

ident este numele programului compilator corespunzător limbajului de programare în care a fost scris programul utilizatorului. Facem aici observația că cu o cartelă COMPILE pot fi compilate una sau mai multe unități de program.

listă de opțiuni conține opțiuni specifice fiecărui compilator; opțiunile sînt despărțite între ele prin virgulă.

Opțiunile pentru compilatorul FORTRAN sînt următoarele:

1) PUN — determină scoaterea pe fișierul de perforare a programului în format B.T. obținut;

2) CRF — permite listarea unui tabel de referințe încrucișate, tabel care are următoarea formă:

REFERENCES CROISEES

var nr nr nr nr ...

var nr nr nr nr ...

.....

unde *var* este numele unei variabile, iar *nr* reprezintă numărul instrucțiunii în care este folosită;

3) NLG — această opțiune suprimă listarea programului sursă (în lipsa acestei opțiuni, listarea se efectuează);

4) SEQ — dacă coloanele 73—80 au fost folosite pentru a numerota cartelele, această opțiune va impune compilato-

rului verificarea ordinii acestora. Compilatorul semnalează orice încălcare a ordinii crescătoare a numerelor din coloanele amintite;

5) CDC — această opțiune permite compilarea condiționată. Ea are efect în momentul întîlnirii unei cartele care conține în primele două coloane caracterele C/. În lipsa opțiunii CDC, cartelele de acest tip au interpretarea cunoscută de cartele de comentarii. În prezența opțiunii CDC, aceste cartele sînt interpretate ca și cînd ar conține instrucțiuni FORTRAN; bineînțeles că în această situație numărul de instrucțiune poate avea maximum 3 cifre și trebuie plasat în coloanele 3—5;

6) OBL — prezența acestei opțiuni determină listarea instrucțiunilor în cod mașină (modulul obiect) generate de compilator plecînd de la unitatea de program compilată.

De exemplu, următoarele instrucțiuni FORTRAN

```
CALL CSUB (N, COMB)
STOP
```

generează următorul cod obiect:

```
LIGNE: 3   019C  0 6 2 0 2A 018C   LD4I,2    4TEMP+4
           01A0  0 0 1 0 2F 0000   BSG,1     CSUB
           01A4  0 7 E 0 0A 0000   LD4,14    %

LIGNE: 4   01A8  0 0 0 0 2A 0000   LD4I,0    0
           01AC  0 0 0 0 2F 0000   BSG,0     I%STOP
```

(instrucțiunile compilate se aflau pe poziția a treia și a patra în cadrul programului);

7) MAP — este opțiunea care permite listarea locului pe care fiecare variabilă îl ocupă în memorie (prin indicarea adresei primului octet al porțiunii de memorie rezervată variabilei);

Scanned with OKEN Scanner

Lista determinată de opțiunea MAP are următoarea formă:

```
ETIQ    ADR.      ETIQ    ADR.
**********        **********
   1    00BC         2    0088

SYMBOLE TYPE CLAS ADRESSE LONG
******************************
COMP          T DUMMY  P
I%SEMDIOL     SP
K             I S 0010 P    4
```

Opțiunile care urmează au efect în timpul executării programului și ne ajută la depanarea lui.

8) SRG — se verifică dacă valorile expresiilor folosite ca indici de tablouri se află între limitele stabilite de programator prin instrucțiunile de declarare din cadrul programului. Dacă un indice depășește valoarea maximă stabilită prin program, este tipărit un mesaj de eroare de forma:

ERREUR: OPT SRG (ADRESSE : XXXXXX) VALEUR D'INDICE INCORECTE

9) PCK — permite verificarea concordanței între argumente și parametri ori de cîte ori este apelat un subprogram. O neconcordanță va duce la tipărirea unui mesaj de forma:

ERREUR: I%VERIF (ADRESSE: XXXXXX) PASSAGE DE PARAMETRES INCORECT.

10) DBG — această opțiune are același efect ca opțiunile CDC, SRG, PCK reunite.

Domeniul de valabilitate al opțiunilor ce apar în cartele COMPILE este format din toate unitățile de program care sînt compilate folosind această cartelă de comandă.

CARTELA STANDBY

Această cartelă permite programatorului să oprească momentan executarea lucrării sale în regiunea afectată.

Scanned with OKEN Scanner

Lucrul poate fi reluat numai prin intervenția operatorului, care relansează execuția folosind comanda RRN.

Forma instrucțiunii de comandă STANDBY este următoarea:

```
•        STANDBY
```

Un exemplu de utilizare poate fi următorul:

```
•        **MONTAȚI DISC 47 PE RD3
•        STANDBY
```

Prin această secvență programatorul este asigurat că reluarea execuției lucrării sale se va face numai după ce operatorul va da răspunsul cuvenit mesajului apărut pe consolă ca urmare a prelucrării cartelei STANDBY. Acest mesaj va urma după afișarea cartelei comentariu care precede cartela STANDBY.

CARTELA EOF

Această cartelă se folosește pentru delimitarea sfîrșitului fișierului de date pe cartele. Forma ei este următoarea:

```
• EOF
```

unde identificatorul EOF ocupă coloanele 2—4.

TIPURI DE RULĂRI SIMPLE

1)— *Compilarea unui program FORTRAN*

```
•        JOB .....
•        COMPILE FORTRAN
    unitate de program 1
    unitate de program 2
    .....
    unitate de program n
•        EOJ
```

Prin această lucrare sînt compilate unitățile de program care compun programul scris în FORTRAN.

Această rulare este indicată în special pentru primele treceri ale programului pe calculator, cînd numărul de erori sintactice este în general mare.

2) — *Rularea completă a unui program FORTRAN*

- JOB .....
- COMPILE FORTRAN

unități de program

- LINK
- RUN

date de intrare

- EOF
- EOJ

Acest tip de rulare realizează atît compilarea, cît și editarea legăturilor și execuția programului. El este indicat de a fi folosit numai în momentul în care programul scris în FORTRAN este corect din punct de vedere sintactic, verificare care se poate face cu ajutorul tipului de lucrare descris anterior.

Folosirea opțiunii CDC a compilatorului.

Vom da un exemplu de alcătuire a unei lucrări în care programul FORTRAN conține linii în care primele două coloane sînt „C/".

Considerăm următorul program FORTRAN:

```
      INTEGER CA
      READ(105,1)N,K
    1 FORMAT(2I2)
      I=CA(N,K)
      WRITE(108,2)N,K,I
    2 FORMAT(' ',3I6)
      STOP
      END
```

```
      FUNCTION CA(N,K)
      INTEGER CA
      CA=1
      DO 1 I=1,K
    1 CA=CA*(N-I+1)
C/    M=1
C/    DO 2 I=1,K
C/  2 M=M*I
C/    CA=CA/M
      RETURN
      END
```

1) — Considerăm următoarea lucrare:

```
•       JOB TEST,AN:CCUB,PN:HOPE
•       COMPILE FORTRAN
   programul de mai sus
•       LINK
•       RUN
10 5
•       EOJ
```

Unitățile de program listate de compilator au forma:

```
1       INTEGER CA
2       READ(105,1)N,K
3     1 FORMAT(2I2)
4       I=CA(N,K)
5       WRITE(108,2)N,K,I
6     2 FORMAT(' ',3I6)
7       STOP
8       END
```


Scanned with OKEN Scanner

```
      1            FUNCTION CA(N,K)
      2            INTEGER CA
      3            CA=1
      4            DO 1 I=1,K
      5          1 CA=CA*(N-I+1)
C     6    C/      M=1
C     7    C/      DO 2 I=1,K
C     8    C/    2 M=M*I
C     9    C/      CA=CA/M
     10            RETURN
     11            END
```

Litera C care apare în fața instrucțiunilor FORTRAN care încep cu caracterele C/ arată că aceste instrucțiuni sînt interpretate drept comentarii. De aceea, valoarea care va fi luată de variabila I din programul principal va fi $A_{10}^5 = 30240$.

2) — Considerăm acum următoarea lucrare:

```
•           JOB TEST,AN:CCUB,PN:HOPE
•           COMPILE FORTRAN,CDC
  programul de mai sus
•           LINK.
•           RUN
.0 5
•           EOJ
```

Unitățile de program listate de compilator vor avea aceeași formă ca mai sus cu excepția faptului că în fața instrucțiunilor care încep cu caracterele C/ va apare litera D în locul literei C. Aceasta se datorește prezenței opțiunii CDC în cartela de comandă COMPILE și indică faptul că instrucțiunile în discuție sînt interpretate ca instrucțiuni efective și nu comentarii ca în cazul precedent.

Scanned with OKEN Scanner

Ca urmare, valoarea care va fi luată de variabila I din programul principal va fi $C_{10}^{5} = 252$.

Această facilitate a compilatorului FORTRAN este foarte utilă în faza de punere la punct a programelor. Astfel în textul programului pot fi introduse instrucțiuni de afișare a valorilor unor variabile, lucru care ușurează urmărirea modului de lucru al algoritmului.

După punerea la punct a programului nu este necesară scoaterea acestor cartele din programul sursă, ci numai înlăturarea opțiunii CDC din cartela COMPILE. Prin aceasta toate cartelele care au în primele două coloane caracterele „C/" vor fi considerate comentarii.

Scanned with OKEN Scanner

Capitolul IV

DESCRIEREA STRUCTURII FIȘIERELOR

NOȚIUNEA DE FIȘIER

Numim *volum* o unitate externă de memorie, cum ar fi o bandă magnetică sau un pachet de discuri.

Fără a avea pretenția de a da o definiție, vom înțelege prin *mulțime de date* o colecție de date externe în raport cu memoria centrală. O mulțime de date poate ocupa o parte a unui volum sau poate ocupa unul sau mai multe volume.

Prin *fișier* înțelegem reprezentarea simbolică a unei mulțimi de date.

Într-o mulțime de date, datele sînt împărțite în grupări fizice distincte numite *blocuri*. Din punct de vedere fizic, fișierul este o mulțime organizată de blocuri. În același timp, blocul este unitatea de transfer între suportul mulțimii de date și memoria internă.

Fiecare bloc constă din subdiviziuni logice numite *articole*. Un bloc poate conține unul sau mai multe articole. Articolul este o grupare de date asociate care poate fi identificată și tratată fie ca o unitate, fie prin datele individuale care o compun. Din punct de vedere logic, fișierul este o mulțime organizată de articole.

Noțiunea de fișier are sens numai în cadrul unui program, pe cînd o mulțime de date are o existență fizică reală, independent de program.

În cele ce urmează vom studia numai fișierele care au ca suport discurile și benzile magnetice.

Scanned with OKEN Scanner

ORGANIZAREA FIȘIERELOR

Deosebim două moduri generale distincte de organizare:

— *organizarea definită*. Această organizare permite tratarea fișierelor din punct de vedere logic, deci la nivelul articolelor;

— *organizarea nedefinită*. Acest mod de organizare permite tratarea fișierelor din punct de vedere fizic, deci la nivelul blocurilor.

În continuare vom vorbi numai despre organizarea definită. În cadrul acestui mod general de organizare, distingem trei organizări diferite și anume:

1) *Organizarea secvențială* corespunde faptului că articolele sînt înregistrate pe suport unul după altul, pe măsura obținerii lor. Acesta este cel mai simplu mod de organizare și poate fi folosit pentru orice tip de suport.

Accesul la fișier poate fi făcut numai secvențial, adică nu se poate ajunge la un articol fără a parcurge toate articolele anterioare lui din fișier. Modul secvențial de organizare este indicat pentru rezolvarea acelor probleme în care este necesară prelucrarea articolelor unul după altul și cînd în cursul unei prelucrări sînt implicate majoritatea articolelor.

O limitare datorită hardware-ului care are implicații importante pentru programator constă în faptul că un articol de pe bandă nu poate fi înlocuit prin altul (deci nu se poate face actualizarea unui fișier); pentru rezolvarea acestei situații este necesară crearea unui nou fișier. Dacă însă fișierul secvențial are ca mediu de memorare un disc magnetic, actualizarea fișierului este posibilă.

2) *Organizarea selectivă*. Acest tip de organizare este posibil numai pe suporturi *adresabile* (de exemplu discuri magnetice) adică suporturi care prin construcția lor permit să ajungem la un anumit loc de pe ele fără a fi necesară parcurgerea pozițiilor anterioare. Fiecare articol este identificat printr-un cod (de exemplu numărul său de ordine) în cadrul fișierului. Cunoscînd lungimea articolelor din fișier, numărul de ordine permite localizarea poziției sale pe suport. Cum suportul

Scanned with OKEN Scanner

este adresabil, putem avea acces direct la articolul dorit. Organizarea selectivă nu este permisă atunci cînd lucrăm în FORTRAN.

3) *Organizarea secvențială indexată.* Articolele sînt aranjate în fișier în ordinea crescătoare a unor „chei" care permit identificarea lor. Accesul la ele poate fi făcut sau secvențial sau direct, prin indicarea cheii. Acest mod de organizare nu poate fi folosit dacă limbajul de programare în care este scris programul este limbajul FORTRAN.

ORGANIZAREA SUPORTURILOR

Un fișier poate ocupa unul sau mai multe volume (în întregime sau parțial). La rîndul său, un volum poate conține unul sau mai multe fișiere. Obținem deci următoarele situații posibile:

1) — monofișier — monovolum
2) — monofișier — multivolum
3) — multifișier — monovolum
4) — multifișier — multivolum.

Menționăm că mai există o structură posibilă și anume cea de *zonă partajată,* despre care vom vorbi în capitolul referitor la biblioteci.

Suporturile magnetice prezintă avantajul că pot fi refolosite în momentul în care se consideră că informația stocată pe ele este perimată. Acest avantaj cere însă din partea utilizatorului o grijă deosebită deoarece poate distruge din greșeală fie fișierele proprii, fie fișierele altor utilizatori.

De aceea pentru fiecare fișier pot exista informații de identificare numite *etichete,* care permit protejarea fișierelor; accesul la informația din fișier devine astfel posibilă numai prin cunoașterea și precizarea corectă a acestor etichete.

Sistemul de operare permite utilizarea simplă a unor etichete standard, dar și posibilitatea ca utilizatorul să-și creeze tipuri proprii de etichete. Există deci două categorii de etichete:

Scanned with OKEN Scanner

— *etichete standard* — aceste etichete sînt create și controlate automat de către sistemul de operare. Formatul și amplasamentul lor sînt de asemenea standardizate;

— *etichete utilizator* — etichete pentru care programatorul este cel care definește locul, formatul și conținutul lor.

În această carte vom folosi numai etichete standard, care oferă avantajul ușurinței de manipulare.

Identificatorii etichetelor standard sînt următorii:

VOL — etichetă de început de volum
EOV — etichetă de sfîrșit de volum
HDR — etichetă de început de fișier
EOF — etichetă de sfîrșit de fișier

Observație: Etichetarea suportului este obligatorie pe disc, dar opțională pe bandă.

STRUCTURA ETICHETELOR STANDARD PENTRU FIȘIERE PE BANDĂ MAGNETICĂ

1) *Eticheta VOL* ocupă 80 de octeți. Ea permite identificarea și protecția volumului.

Principalele informații conținute în ea sînt identificatorul volumului, numele proprietarului și un indicator de acces. Aceste informații sînt precizate de utilizator prin intermediul cartelelor de comandă, atunci cînd dorește să utilizeze fișierul.

Indicatorul de acces arată dacă accesul la volum este liber sau este protejat printr-o parolă numită „cuvînt de acces".

Identificatorul de volum este dat prin argumentul VS, numele programatorului prin argumentul ON, iar cuvîntul de acces prin argumentul AC; toate aceste argumente apar în cartela de comandă INIT.

Structura etichetei VOL este următoarea:

Octeți	Conținut
0—3	VOL1 (identificatorul etichetei)
4—9	identificatorul volumului (6 caractere alfanumerice)

Octeți	Conținut
10	indicator de acces. Dacă este un caracter alfanumeric, accesul la fișier este posibil numai prin indicarea cuvîntului de acces specificat la crearea fișierului. Dacă este un blanc, accesul nu este condiționat de indicarea cuvîntului de acces corect
11—36	rezervat
37—50	numele proprietarului (14 caractere alfanumerice)
51—78	rezervat
79	cifra 1 (indicator de etichete standard)

2) *Eticheta HDR1* ocupă 80 de octeți, conține informații despre fișier și volumul (volumele) pe care acesta le ocupă. Structura ei este următoarea:

Octeți	Conținut
0—3	HDR1 (identificatorul etichetei)
4—20	identificatorul fișierului (17 caractere alfanumerice)
21—26	identificatorul ansamblului de volume (pentru structura multivolum)
27—30	număr zecimal care reperează volumele succesive din fișier
31—34	număr zecimal care reperează fișierele succesive dintr-un multifișier mono- sau multivolum
35—38	numărul de generație al fișierului (în zecimal)
39—40	numărul de versiune (în zecimal)
41—46	data de creare, are forma baazzz unde aa indică anul, iar zzz ziua.
53	indicator de acces
54—59	000000
60—79	rezervat

Numărul de generație permite să facem distincție între două fișiere cu același nume.

Numărul de versiune permite să facem distincție între două fișiere cu același nume și același număr de generație.

Scopul acestor două informații este de a putea face modificări într-un fișier în sensul creării unui nou fișier cu același

Scanned with OKEN Scanner

nume, păstrînd însă și vechea formă pentru utilizatorii care doresc să lucreze cu aceasta.

Eticheta HDR2 descrie structura informațiilor din fișier. Lungimea etichetei este de 80 de octeți. Structura ei este următoarea:

Octeți	Conținut
0—3	HDR2 (identificatorul etichetei)
4	formatul articolelor din fișier (F pentru format fix, V pentru format variabil, U sau orice alt caracter diferit de F și V pentru format nedefinit)
5—9	lungimea blocului (în zecimal)
10—14	lungimea articolului
15—49	rezervat
50—51	fără semnificație în acest manual
59—79	rezervat

Etichetele EOF1 și *EOV1*

Octeți	Conținut
0—3	EOF1 sau EOV1 după cum este vorba de sfîrșit de fișier sau sfîrșit de volum
4—53	la fel ca în HDR1
54—59	numărul de blocuri din secțiunea fișierului conținută în volum (în zecimal)
60—79	la fel ca în HDR1

Etichetele EOF2 și *EOV2*

Octeți	Conținut
0—3	EOF2 sau EOV2 după cum este vorba de sfîrșit de fișier sau sfîrșit de volum
4—79	la fel ca în HDR2

Eticheta HDR conține în ordine etichetele HDR1 și HDR2.

Eticheta EOF conține în ordine etichetele EOF1 și EOF2.

Eticheta EOV conține în ordine etichetele EOV1 și EOV2.

Pentru delimitarea informațiilor de natură diferită, pe bandă se introduc informații speciale numite mărci de fișier (MF).

Scanned with OKEN Scanner

Structura unui fișier secvențial pe bandă este prezentată în cele ce urmează:

Mono fișier — monovolum

V	H	M		M	E	M	M
O	D		Fișier		O		
L	R	F		F	F	F	F

Mono fișier — multivolum

Primul volum și volumele intermediare

V	H	M		M	E	M	M
O	D		Fișier		O		
L	R	F		F	V	F	F

Ultimul volum

V	H	M		M	E	M	M
O	D		Fișier		O		
L	R	F		F	F	F	F

Multi fișier — monovolum

V	M	M		M	E	M	H	M				M	E	M	M
O	D		Fișier		O		D		Fișier	-----	Fișier		O		
L	R	F		F	F	F	R	F				F	F	F	F

Multifișier — multivolum

Primul volum:

V	H	M		M	E	M	H	M		M	E	M	M
O	D		Fișier A		O		D		Fișier B		O		
L	R	F		F	F	F	R	F		F	V	F	F

Scanned with OKEN Scanner

Volumele intermediare

V O L	H D R	M F	Fișier B (continuare)	M F	E O F	M F	H D R	M F	Fișier C	E O V	M F	M F

Ultimul volum

V O L	H D R	M F	E O F	M F	H D R	M F	Fișier D	M F	E O F	M F	M F

Observăm că pentru fișierul C sfîrșitul fișierului a coincis cu sfîrșitul volumului; în această situație este înscrisă eticheta de sfîrșit de volum, urmînd ca etichetele de sfîrșit de fișier să fie înscrise la începutul volumului următor.

STRUCTURA ETICHETELOR STANDARD PENTRU FIȘIERE PE DISC

1) *Eticheta VOL* are lungimea de 64 octeți, informația pe care o conține fiind următoarea:

Octeți	Conținut
0—3	VOL1 (identificatorul etichetei)
4—9	identificatorul volumului
10	indicator de acces
11—24	numele proprietarului
25—63	rezervate

2) *Eticheta HDR1* are lungimea de 96 octeți. Conținutul ei este următorul (prin fișier se va înțelege și zonă partajată):

Octeți	Conținut
0—3	HDR1
4—20	identificatorul fișierului
21—26	identificatorul ansamblului de volume

Scanned with OKEN Scanner

Octeți	Conținut
27—28	numărul de ordine al volumului în fișier (în zecimal)
29—32	numărul de generație (în zecimal)
33—34	numărul de versiune (în zecimal)
35—39	data de creare a fișierului (aazzz)
40—44	data de expirare a fișierului (aazzz)
45	indicator de acces
46—48	adresa disc absolută a etichetei HDR2 (în binar)
49	numărul de cilindri afectați (în binar)
50	codul ultimei etichete HDR1 a fișierului (în bitul 0), și numărul de zone din etichetă (biții 4—7)
51—53	adresa disc absolută a următoarei etichete HDR1
54	numărul zonei (în binar)*
55—57	adresa disc absolută a începutului zonei (în binar)
58—60	adresa disc absolută a sfîrșitului zonei (în binar)
61	numărul zonei de extensie (în binar)*
62—64	adresa disc absolută a zonei de extensie (în binar)*
65—67	adresa disc absolută a sfîrșitului zonei de extensie (în binar)*
68—95	rezervat

* Dacă zona este afectată fișierului dar nu a fost încă folosită, numărul ei este majorat cu 128 (10000000 în binar)

3) *Eticheta HDR2* ocupă 31 octeți. Conținutul ei diferă după cum exploatarea se face secvențial (ca pe bandă) sau în acces direct (ne mărginim numai la posibilitățile de exploatare a fișierelor oferite de FORTRAN)
Acces secvențial:

Octeți	Conținut
0—3	HDR2
4	C
5—6	dimensiunea unei pagini (partea utilă a blocului) exprimată în binar
7	formatul articolelor (F pentru format fix, V pentru format variabil și U pentru format nedefinit)
8—9	lungimea unui articol

Scanned with OKEN Scanner

Octeți	Conținut
10—12	adresa disc relativă a primului bloc din fișier
13—15	adresa disc relativă a ultimului bloc din fișier (numai pentru ultima etichetă HDR2 a fișierului)
16	nesemnificativ în acest manual
17—31	rezervat

Acces direct:

Octeți	Conținut
0—3	HDR2
4	U
5—6	dimensiunea unei pagini (în binar)
7—31	rezervat

În cazul unei zone partajate:

Octeți	Conținut
0—3	HDR2
4	P
5—6	dimensiunea unei pagini (în binar)
7	rezervat
8—9	adresa disc relativă a tabelei de alocare
10—14	rezervat
15—17	adresa disc absolută a primei etichete de fișier membru al zonei partajate
18	fără semnificație în această carte
19	numărul de sectoare ocupate de un bloc (în binar)
20	numărul de blocuri
21—31	rezervat

4) *Eticheta HDR3* este folosită numai pentru zonele partajate și conține informații asupra fișierelor membre ale zonei partajate.

Octeți	Conținut
0—3	HDR3
4—20	identificatorul fișierului membru

Scanned with OKEN Scanner

Octeți	Conținut
21	indicele de actualizare (în binar)
22—23	adresa disc relativă a primului bloc din fișier
24—25	adresa disc relativă a ultimului bloc din fișier
26	formatul articolelor
27—28	lungimea articolului (în binar)
29—31	adresa disc absolută a următoarei etichete HDR3 a zonei partajate

Toate aceste etichete descrise mai sus apar pe cilindrul 0 al discului și formează *zona de informații de control* (ZIC). Informația din această zonă este organizată după cum urmează:

sectorul 0 — rezervat pentru sistem

sectorul 1 — primii 64 octeți conțin identificatorul volumului

sectorul 2 — primii 200 octeți formează tabela de ocupare a volumului, fiecare octet conținînd $(00)_{16}$ sau $(01)_{16}$ după cum cilindrul corespunzător este sau nu disponibil.

sectoarele 3—34 — conțin tabela etichetelor HDR1 și HDR2. Fiecare sector poate conține două perechi de etichete HDR1 și HDR2. Sectoarele sînt utilizate în ordinea 3, 5, 7,...,33, 4, 6...,32, 34

sectorul 35 — rezervat pentru sistem

sectoarele 36—119 — conțin tabela etichetelor HDR3. Un volum poate conține cel mult 5 zone partajate, iar o zonă partajată poate fi formată din cel mult 128 fișiere membre. Sectorul 36 conține tabela de control a zonelor partajate; are 5 intrări corespunzătoare celor 5 zone posibile; fiecare intrare are un indi-

cator de afectare a zonei (1 octet) și adresa primei etichete HDR3 corespunzătoare zonei (3 octeți). Sectoarele 37—119 conțin etichetele HDR3 (cîte 8 etichete în fiecare sector)

Informații suplimentare despre etichete și organizarea lor vor fi date în capitolul referitor la programe utilitare, unde printre altele vor fi expuse posibilități de listare a acestor etichete.

Scanned with OKEN Scanner

Capitolul V

EXPLOATAREA FIȘIERELOR ÎN FORTRAN

În linii mari, principiile generale care stau la baza exploatării fișierelor sînt valabile și atunci cînd lucrăm cu limbajul de programare FORTRAN. Totuși, datorită limitărilor care au fost făcute în momentul creării limbajului, datorită dorinței de a-l păstra cît mai stabil cu putință și datorită largii sale utilizări, în FORTRAN nu sînt valabile toate tipurile de organizare a datelor posibile pentru sistemul FELIX.

În FORTRAN există două moduri posibile de organizare a fișierelor:

— organizarea nedefinită FORTRAN
— organizarea secvențială

și două moduri de utilizare a conținutului acestora:

— exploatare secvențială
— acces direct.

În cazul fișierelor pe bandă distingem:

— organizarea secvențială cu acces secvențial
— organizarea nedefinită FORTRAN cu acces secvențial.

În cazul fișierelor pe disc există următoarele posibilități:

— organizarea secvențială cu acces secvențial
— organizarea nedefinită FORTRAN cu acces secvențial
— organizarea nedefinită FORTRAN cu acces direct.

După cum vom vedea în cadrul acestui capitol, pentru fișiere pe bandă și disc exploatate secvențial, modul de organizare este secvențial sau nedefinit FORTRAN după cum în ordinul DEFINE FILE apare sau nu opțiunea SFD.

Pentru a face acest capitol de sine stătător, vom începe prin a prezenta noțiunile de FORTRAN necesare.

PRINCIPALELE INSTRUCȚIUNI FORTRAN PENTRU EXPLOATAREA FIȘIERELOR

În acest paragraf nu vom epuiza lista instrucțiunilor FORTRAN care pot fi utilizate la exploatarea fișierelor și nu vom expune toate posibilitățile lor, mărginindu-ne doar la cele folosite în mod curent. O prezentare exhaustivă poate fi găsită de exemplu în manualul FORTRAN al firmei.

Instrucțiuni pentru fișiere exploatate secvențial

1) *Instrucțiunea FORMAT*

Forma acestei instrucțiuni este următoarea:

FORMAT (*listă de formate*)

Instrucțiunea FORMAT are totdeauna un număr de instrucțiune. Vom presupune cunoscute formatele care pot apare în listă și modul în care ele se pot grupa. Vom descrie aici numai modul în care această instrucțiune definește înregistrări și cum se stabilește legătura între lista de date și lista de formate.

La compilare, lista de formate este desfăcută, eliminîndu-se parantezele și factorii de repetiție. În același timp se marchează poziția parantezei deschise corespunzătoare ultimei paranteze închise; dacă în lista de formate nu apar paranteze, marcajul este plasat la începutul listei.

Reamintim că formatele se împart în formate de date (F, E, G, D, I, L, Z, A) și formate de control (H, X, T, formatul literal)

Instrucțiunea FORMAT definește înregistrările în modul următor: O înregistrare cuprinde formatele situate între:

— începutul listei și primul slash
— două slashuri consecutive
— ultimul slash și sfîrșitul listei
— începutul și sfîrșitul listei (numai dacă în lista de formate nu apar slashuri).

Corespondența între liste de date care apare într-o instrucțiune de citire sau scriere și lista de formate din instrucțiunea FORMAT conform căreia se face citirea sau scrierea se stabilește după cum urmează: se parcurg pe rînd datele din lista

Scanned with OKEN Scanner

de date și formatele din lista de formate; dacă unei date îi corespunde un format de date, data respectivă va fi transmisă conform acelui format; dacă datei îi corespunde un format de control, se va întreprinde acțiunea indicată de acesta și datei îi va fi pus în corespondență formatul următor, lucrurile repetîndu-se în funcție de tipul acestuia. Procesul se repetă pînă la epuizarea listei de date.

Dacă în lista de date numărul datelor este mai mare decît numărul formatelor de date din lista de formate, aceasta din urmă este reluată de la marcajul stabilit anterior, determinîndu-se în același timp începutul unei noi înregistrări.

Exemplu: Să presupunem că lista de date este formată din primele 7 componente ale vectorului întreg A și că lista de formate din instrucțiunea FORMAT corespunzătoare are forma:

2(I2,2X)/I3,1X,2(I3,2X)

Desfăcută, această listă capătă forma:

I2,2X,I2,2X/I3,1X,I3,2X,I3,2X

Atunci vor lua naștere următoarele înregistrări:

Înregistrare	*Lungime*	*Datele cuprinse în înregistrare*
I2, 2X, I2, 2X	8	A(1), A(2)
I3, 1X, I3, 2X, I3, 2X	14	A(3), A(4), A(5)
I3, 2X, I3, 2X	10	A(6), A(7)

2) *Instrucțiunea READ cu format*

Forma instrucțiunii este:

READ(*n*,*f* [,END=*n1*] [,ERR=*n2*]) *listă de date*

listă de date — lista datelor care trebuie scrise în fișier

n — numărul de referință al fișierului în cadrul programului FORTRAN (despre el se va vorbi la prezentarea ordinului DEFINE FILE)

f — numărul de instrucțiune al instrucțiunii FORMAT conform căruia se face transmiterea datelor

n1, *n2* numere de instrucțiuni.

Scanned with OKEN Scanner

Opțiunea END, dacă este prezentă, are ca efect trecerea controlului programului la instrucțiunea cu numărul de instrucțiune *n1* dacă s-a detectat sfîrșitul fișierului (vezi instrucțiunile ENDFILE și REWIND), evitîndu-se astfel terminarea programului în situația respectivă.

Opțiunea ERR permite evitarea terminării programului în cazul în care apare o eroare de transmitere care nu a putut fi corectată de sistemul de exploatare; prezența ei face ca în asemenea situații controlul programului să fie trecut instrucțiunii cu numărul de instrucțiune *n2*.

3) *Instrucțiunea WRITE cu format*

WRITE *(n, f) listă de date*

Semnificația parametrilor este aceeași ca în cazul instrucțiunii READ cu format.

4) *Instrucțiunile READ și WRITE fără format*

Forma lor este următoarea:

READ *(n* [, END=*n1*] [, ERR=*n2*]) *listă de date*

WRITE *(n) listă de date*

În cazul acestor instrucțiuni, înregistrarea este definită de datele din lista de date; deci instrucțiunile de mai sus definesc o singură înregistrare.

5) *Instrucțiunea BACKSPACE* are forma:

BACKSPACE *n*

În cazul fișierelor secvențiale pe disc sau bandă, în care articolele sînt neblocate, efectul instrucțiunii este poziționarea fișierului:

— înaintea ultimei înregistrări FORTRAN citită sau scrisă (dacă la creare a fost folosită scrierea cu FORMAT);

— înaintea tuturor înregistrărilor FORTRAN create de același WRITE fără format (dacă ultima înregistrare citită sau scrisă înainte de BACKSPACE a fost înregistrare fără format).

În cazul în care articolele sînt blocate (format fix blocat sau variabil blocat) poziționarea se face înaintea ultimului

Scanned with OKEN Scanner

bloc citit sau scris (vom vedea că un bloc este scris în momentul în care spațiul rămas disponibil în tampon este mai mic decît lungimea maximă a unui articol).

Citirea etichetei de sfîrșit de fișier (prin READ cu opțiunea END) și folosirea lui BACKSPACE permite scrierea în continuare în fișier.

Dacă fișierul este la începutul său, efectul instrucțiunii este nul.

6) *Instrucțiunea ENDFILE are* forma:

ENDFILE *n*

și executarea sa are ca efect scrierea etichetei de sfîrșit de fișier pe suportul fișierului cu numărul de referință *n*, delimitîndu-se astfel sfîrșitul fișierului.

7) *Instrucțiunea REWIND* are forma:

REWIND *n*

și are ca efect poziționarea fișierului la începutul său. În plus, dacă ultima operație a fost o scriere în fișier, executarea instrucțiunii REWIND face ca pe suportul fișierului să fie înscrisă eticheta de sfîrșit de fișier, după care fișierul este poziționat la începutul său.

Facem observația că după executarea unei instrucțiuni ENDFILE, pentru a folosi în continuare fișierul, el trebuie să fie redeschis. Aceasta se realizează printr-o instrucțiune READ, care are ca efect poziționarea fișierului la începutul său, citirea făcîndu-se deci de la începutul fișierului. În consecință, nu este nevoie să folosim instrucțiunea REWIND după ENDFILE; totuși este bine să o folosim dacă următoarea citire din fișier nu se execută imediat, deoarece în această situație fișierul este repoziționat la începutul său simultan cu executarea de instrucțiuni care nu afectează fișierul.

Menționăm de asemenea că la sfîrșitul unui program FORTRAN toate fișierele sînt închise, iar cele de pe bandă sînt repoziționate la începutul lor.

Scanned with OKEN Scanner

Instrucțiuni pentru fișiere exploatate în acces direct

1) *Instrucțiunile READ și WRITE*

Forma lor este următoarea:

READ *(n'e*[,*v*] [,*f*] [,*ERR=n2*]) *listă de date*

WRITE *(n'e* [,*v*] [,*f*]) *listă de date*

unde *n*, *f*, *n2* au semnificațiile cunoscute.

e — expresie întreagă a cărei valoare determină numărul articolului din fișier care urmează să fie citit;

v — variabilă întreagă care la sfîrșitul operației de citire sau scriere are ca valoare numărul articolului imediat următor ultimului articol citit, respectiv scris.

2) *Instrucțiunea FIND* are forma

FIND *(n'e)*

Efectul ei este poziționarea dispozitivului de citire/ scriere al discului la cilindrul care conține articolul cu numărul dat de valoarea expresiei întregi *e*.

Folosirea instrucțiunii FIND poate duce la micșorarea timpului de execuție al programului deoarece executarea ei se face concomitent cu executarea de alte instrucțiuni care nu afectează fișierul; astfel un READ sau WRITE ulterior referitor la articolul respectiv găsește fișierul gata poziționat la acest articol.

FORMATELE ARTICOLELOR

Într-un fișier exploatat cu ajutorul limbajului FORTRAN, articolele pot avea următoarele formate:

1) *format nedefinit;*

2) *format fix (neblocat)*: articolele sînt de lungime fixă; fiecare bloc conține cîte un articol;

3) *format fix blocat*: articolele sînt de lungime fixă; fiecare bloc conține același număr întreg de articole;

4) *format variabil (neblocat)*: articolele sînt de lungime variabilă; un bloc conține un singur articol;

5) *format variabil blocat:* articolele sînt de lungime variabilă, fiecare bloc conține un număr întreg *n* de articole.

Scanned with OKEN Scanner

După cum vom vedea tot în acest capitol, pentru a putea lucra cu fișiere, utilizatorul trebuie să declare explicit aceste fișiere în programul său. În declarare trebuie specificate lungimea articolului și lungimea blocului. Ele se măsoară în octeți. Aceste lungimi diferă în funcție de modul de exploatare a fișierului și de formatul articolelor.

STRUCTURA BLOCULUI ȘI ARTICOLULUI ÎN FIȘIERE EXPLOATATE SECVENȚIAL

Lucrurile se prezintă diferit după cum la crearea fișierelor s-a folosit scrierea cu ajutorul instrucțiunii FORMAT sau fără ajutorul acesteia.

Scrierea cu instrucțiunea FORMAT

1) *format nedefinit*

Un bloc este format dintr-un singur articol.

Lungimea articolului este egală cu lungimea maximă a înregistrărilor.

Lungimea blocului este egală cu lungimea articolului (dacă suportul fișierului este o bandă magnetică) sau este egală cu lungimea articolului +8 (dacă suportul este un disc); cei 8 octeți suplimentari sînt rezervați pentru antetul blocului.

Facem observația că dacă la scriere lungimea înregistrării este mai mică decît lungimea articolului, în articol este înscrisă numai partea utilă.

2) *format fix*

Un bloc este format dintr-un singur articol.

Articolul conține înregistrarea (de lungime fixă); lungimea lui coincide cu lungimea înregistrării.

Lungimea blocului este egală cu lungimea articolului (cînd suportul este o bandă magnetică) sau egală cu lungimea articolului +8 (cînd suportul este un disc).

Facem observația că dacă lungimea datelor care se scriu în fișier este mai mică decît lungimea înregistrării, articolul este completat la dreapta cu blancuri.

Scanned with OKEN Scanner

3) *format fix blocat*

Un bloc conține un număr întreg n de articole.

Lungimea articolului este egală cu lungimea (fixă) a înregistrărilor.

Lungimea blocului este de n ori lungimea articolului (pentru bandă magnetică) sau de n ori lungimea articolului +8 (pentru disc).

Conținutul tamponului este înscris într-un bloc al fișierului în momentul în care el conține n înregistrări.

4) *format variabil*

Articolul este format din antetul articolului (4 octeți) și din înregistrarea FORTRAN. Primii doi octeți ai antetului articolului conțin lungimea părții scrise a articolului.

Lungimea articolului este egală cu lungimea maximă a unei înregistrări +4.

Un bloc conține un singur articol. Acesta este precedat în bloc de antetul blocului (care ocupă 8 octeți pentru disc și 4 octeți pentru bandă), care în primii săi doi octeți conține lungimea părții scrise a blocului. Deci, lungimea blocului este egală cu lungimea articolului la care se adaugă lungimea antetului blocului.

Structura blocului este următoarea:

Antetul blocului	Antetul articolului	Articol

5) *format variabil blocat*

Structura unui articol și lungimea sa sînt aceleași ca în cazul formatului variabil.

Blocul este format dintr-un număr întreg n de articole precedate de antetul blocului (a cărui lungime este de 4 octeți pe bandă și 8 octeți pe disc). Lungimea blocului este egală cu lungimea antetului blocului plus de n ori lungimea articolului.

Structura unui bloc este următoarea:

Antetul blocului	Antetul articolului	Articol	----------	Antetul articolului	Articol

Scanned with OKEN Scanner

Scrierea fără FORMAT

Acest mod de scriere este posibil numai în fișiere în care formatul articolelor este variabil sau variabil blocat.

Un articol este format din antetul articolului (4 octeți) și partea folosită efectiv pentru scrierea informațiilor în fișier. În antetul articolului, primul octet memorează lungimea articolului, iar biții 23—24 au următoarea semnificație:

00 — este vorba de primul și ultimul articol din bloc
01 — primul articol din bloc
10 — ultimul articol din bloc
11 — este vorba de un articol care nu este nici primul nici ultimul din bloc.

Un bloc este constituit din antetul blocului (4 octeți pe bandă și 8 octeți pe disc) și un articol (pentru formatul variabil) sau *n* articole (pentru formatul variabil blocat); lungimea blocului se calculează corespunzător.

Informația care trebuie scrisă cu o aceeași instrucțiune WRITE fără format este scrisă începînd cu un nou bloc; ea este împărțită în segmente de lungime egală cu lungimea articolului -4, segmente care sînt înscrise succesiv în articolele fișierului.

STRUCTURA BLOCULUI ȘI ARTICOLULUI PENTRU FIȘIERE EXPLOATATE ÎN ACCES DIRECT

Singurul format permis pentru asemenea fișiere este formatul fix blocat.

Lungimea articolului este deci egală cu lungimea unei înregistrări, iar lungimea blocului este egală cu de *n* ori lungimea articolului (blocul nu este precedat de vreun antet).

Reamintim faptul că un cilindru are 120 de sectoare și că un sector are 256 octeți (discuri DIAM) sau 1024 octeți (discuri DIMAS).

Scanned with OKEN Scanner

Un bloc începe totdeauna de la începutul unui nou sector și se poate întinde pe unul sau mai multe sectoare. Numărul de sectoare ocupate de un bloc trebuie să dividă pe 120 (pentru ca un bloc să nu se întindă pe doi cilindri). Acest lucru cade în sarcina utilizatorului.

DECLARAREA FIȘIERELOR ÎNTR-UN PROGRAM FORTRAN

Oricărui fișier pe care îl folosim trebuie să-i atașăm un *idex* (identificator de exploatare) și unul sau mai multe *numere de referință*. Numărul de referință ne permite să precizăm în instrucțiunile de intrare/ieșire despre ce fișier este vorba.

Idexul fișierelor este folosit în cartelele de comandă pentru a completa informațiile despre fișier. Cartelele de comandă referitoare la fișiere vor fi prezentate în capitolul următor.

Ordinul DEFINE FILE stabilește legătura între idexul fișierului și numerele sale de referință și permite introducerea în program a unor informații referitoare la organizarea sa; aceste informații sînt folosite de compilator care alcătuiește tabele de descriere a fișierelor (TDF) pe baza cărora se pot efectua operațiile de intrare/ieșire.

Ordinul DEFINE FILE trebuie plasat după cartela COMPILE și anume în fața programului principal. Forma sa este următoarea:

* DEFINE FILE *listă de argumente*

În scrierea lui trebuie respectate următoarele reguli sintactice:

— coloana 1 conține un asterisc
— coloanele 2 și 14 conțin cîte un blanc
— coloanele 3—13 conțin identificatorul DEFINE FILE
— lista de argumente începe în coloana 15; poate continua pînă la coloana 71. Este permisă folosirea de cartele de continuare, care sînt semnalate prin prezența unui asterisc în coloana 72 a cartelei anterioare. O cartelă de continuare conține un asterisc în prima coloană, blancuri în co-

Scanned with OKEN Scanner

loanele 2—11 și continuarea informațiilor de pe cartela precedentă în coloanele 12—71. Argumentele din listă sînt despărțite între ele prin virgulă.

În general, în lista de argumente nu pot apare blancuri. Această regulă generală poate fi încălcată doar pentru a evita despărțirea unui argument în două părți situate pe cartele diferite; în această situație se permite ca ultima virgulă de pe o cartelă să fie urmată de blancuri pînă la coloana 71 inclusiv.

Argumentele de ordinul DEFINE FILE diferă după cum fișierul care trebuie declarat este un fișier al sistemului sau un fișier al utilizatorului.

1) *Fișierele sistemului*

Fișierul de lucrări are idexul *1 și are atașat automat numărul de referință 105.

Fișierul de ieșire are idexul *2 și are atașat automat numărul de referință 108.

Fișierul de ieșire prin perforare are idexul *3 și numărul de referință 106, de asemenea atașat automat.

Fișierul LOADGO are idexul *4. Lui nu i se atașează un număr de referință nici automat, nici prin intermediul ordinului DEFINE FILE deoarece nu avem acces la el prin intermediul unui program scris în FORTRAN.

Pentru fișierele sistemului cu idexurile *1, *2, *3, rolul ordinului DEFINE FILE este de a atașa și alte numere de referință, existînd posibilitatea de a anula sau nu pe cele atașate automat. Un număr de referință atașat automat este anulat atunci cînd apare în ordinul DEFINE FILE ca număr de referință al unui alt fișier; în caz contrar, el este păstrat.

În cazul fișierelor sistemului, argumentele din DEFINE FILE au următoarea formă:

$$idex = n_1 = n_2 = \ldots = n_p$$

unde $n_1, n_2, \ldots, n_p$ sînt numere naturale mai mici decît numărul 2^{31} și reprezintă numerele de referință atașate fișierului cu idexul *idex*. În cadrul programului FORTRAN ne putem referi la fișier prin intermediul oricăruia dintre numerele de referință atașate.

2) Fișierele utilizatorului

Pentru aceste fișiere nu există numere de referință atașate automat. Forma argumentelor din ordinul DEFINE FILE este în acest caz următoarea:

an(DVT: *perif*[,RCF:*r*][,RCS:*s*][,BFS:*b*][,SFD])=n_1 ... =n_p

unde *a* este un caracter alfabetic reprezentînd idexul fișierului, iar *n* este una dintre cifrele 0, 1, 2, avînd următoarea semnificație:

n = 0 — fișierul este exploatat atît prin citire cît și prin scriere

n = 1 — fișierul este exploatat numai prin citire

n = 2 — fișierul este exploatat numai prin scriere.

Informațiile n_1, n_2, ..., n_p reprezintă numerele de referință ale fișierului. Ele sînt numere naturale mai mici decît 2^{31} și cu ajutorul lor putem face referiri la fișier în instrucțiunile de citire/scriere.

perif descrie tipul suportului și poate fi (în cadrul restrictiv considerat în acest manual):

MT — pentru bandă magnetică
RD — pentru disc DIAM
MD — pentru disc DM25
AD — pentru disc MD50

r indică formatul articolelor și trebuie specificată după cum urmează:

D — fișier pe disc exploatat în acces direct
U — format nedefinit
F — format fix
FB — format fix blocat
V — format variabil
VB — format variabil blocat

Dacă informația RCF: *r* nu este specificată, se consideră *r* = U;

s reprezintă lungimea articolului și ea trebuie specificată atunci cînd *r* este D, FB sau VB;

b reprezintă lungimea blocului; el este un număr natural mai mic decît 65536. Prin lipsă, *b* = 256.

Scanned with OKEN Scanner

Opțiunea SFD se referă numai la fișiere pe disc sau bandă exploatate secvențial. Prezența acestei opțiuni face ca organizarea fișierului să fie secvențială; în lipsa opțiunii SFD, fișierul va avea organizarea nedefinită FORTRAN.

Un fișier cu organizare secvențială poate fi folosit sau numai pentru scriere, sau numai pentru citire; pentru un astfel de fișier scrierea se poate face fără FORMAT numai dacă fiecare instrucțiune WRITE sau READ determină scrierea sau citirea unui singur articol.

Observații finale asupra fișierelor utilizatorului

— pentru fișierele pe bandă sînt necesare cartelele de comandă INIT, ASSIGN și LABEL;

— pentru fișierele pe disc sînt necesare cartelele de comandă INIT, ASSIGN, LABEL și eventual ALLOC.

Scanned with OKEN Scanner

Capitolul VI

CARTELELE DE COMANDĂ PENTRU FIȘIERE

În conformitate cu capitolele precedente, ne vom referi numai la fișiere pe benzi și discuri magnetice. Cartelele de comandă pe care le vom prezenta permit efectuarea următoarelor operații:

1) — asignare (cartela ASSIGN);
2) — inițializare a unui volum (cartela INIT);
3) — alocare de spațiu pentru fișier;
4) — identificare (cartela LABEL).

Rezervarea unei arii pentru fișier se poate face sau de către sistemul de gestiune al fișierelor sau de către monitor. În primul caz folosim un număr de argumente în cartela LABEL, iar în al doilea caz folosim cartela ALLOC.

Toate cele patru cartele amintite trebuie să apară înainte de cartela RUN. Cartela INIT, dacă apare, trebuie să fie prima referitoare la fișier.

CARTELA ASSIGN

Prin intermediul acestei cartele se stabilește o corespondență între un fișier (identificat prin idexul său) și unul sau mai multe periferice care reprezintă suportul său. Forma cartelei este următoarea:

• ASSIGN *a*, DV: *perif* s *perif* s ... s *perif* [,HOLD]

a este idexul fișierului. Este constituit dintr-o literă.

Scanned with OKEN Scanner

perif are forma *tip n* unde:

tip este { RD pentru discuri DIAM
MD pentru discuri MD25
AD pentru discuri MD50
MT pentru bandă

n este un număr natural mai mic decît 100 reprezentînd numărul unității periferice;

s este un separator care poate fi + sau / avînd semnificațiile de „și", respectiv „și (dacă este posibil)".

Cartela ASSIGN se referă la un singur fișier. Corespondența între fișier și periferice stabilită prin ASSIGN este păstrată doar pe durata unei faze de execuție (dacă nu apare argumentul HOLD) sau pe durata întregii lucrări (dacă apare argumentul HOLD).

Periferice citate într-o cartelă ASSIGN trebuie să fie toate de același tip.

CARTELA LABEL

Această cartelă definește identitatea fișierului (definind parametrii de creare sau exploatare) și eventual (numai pentru discuri) precizează condițiile de alocare a spațiului. Forma ei este următoarea:

• LABEL *a*,FN:'*nume*'[,GN:*g*][,VN:*v*][,CD:*dată*]
[,RP:*r*][,FS:*f*][,AM:{ANY / NEW / OFL}][,SZ:*s*][,ON:'*proprietar*']
[,FP:'*parolă*'] [,VP:'*parolă*'][,OV:*v*][,{OD: *dată* / OG: *g*}]

Semnificația parametrilor este:

a — idexul fișierului;

nume — numele fișierului; este compus din 1—17 caractere alfanumerice;

g — numărul de generație al fișierului; este un număr natural mai mic decît 10000 și mai mare ca zero; prin lipsă, $g = 1$;

v — numărul de versiune al fișierului; este un număr natural mai mic decît 100 și mai mare sau egal cu zero; prin lipsă, $v = 0$;

dată — reprezintă data creării fișierului. Are forma *aazzz* unde *aa* reprezintă anul, iar *zzz* reprezintă ziua;

r — perioada de rezervare a fișierului. Este un număr natural mai mic decît 1000 și folosește la evitarea distrugerii fișierului înainte de *r* zile de la creare. Dacă nu este specificat, se consideră $r = 0$.

f — este un identificator compus din șase caractere alfanumerice și care este comun tuturor fișierelor de pe același volum. Prin lipsă, este format din șase blancuri;

proprietar — numele proprietarului (format din 1—14 caractere);

parolă — șir de 1—10 caractere reprezentînd un cuvînt de recunoaștere al fișierului (în opțiunea FP) sau un cuvînt de recunoaștere a volumului (în opțiunea VP).

Argumentele AM și SZ se folosesc dacă vrem ca alocarea de spațiu pentru fișier să fie făcută, în momentul creerii acestuia, de către sistemul de gestiune al fișierelor. Dacă lipsește, se presupune că alocarea de spațiu se face de către monitor (prin cartela ALLOC). Opțiunile care apar în argumentul AM sînt:

ANY — orice porțiune liberă de pe disc poate fi utilizată;
NEW — trebuie folosit un disc în întregime liber;
OFL — trebuie folosită o generație dată a aceluiași fișier (precizată prin opțiunile OD, OG, OV).

În argumentul SZ apare informația *s* care reprezintă numărul de cilindri alocați fișierului; este un număr pozitiv mai mic decît 65 536.

În concluzie, la creare, cartela LABEL atribuie fișierului un nume, o perioadă de rezervare și eventual numere de generație și de versiune. În plus, în cazul în care suportul fișierului este un disc, se poate aloca spațiu pentru fișier. La exploatarea fișierului, cartela LABEL permite identificarea acestuia.

Scanned with OKEN Scanner

CARTELA INIT

Această cartelă permite inițializarea (premarcarea) unui volum (bandă sau disc). Forma ei este următoarea:

- INIT DV:*perif*,VS:*serie*[,FS:*f*]

unde *serie* este un șir de șase caractere alfanumerice reprezentînd identificatorul volumului.

CARTELA ALLOC

Această cartelă este utilizată pentru a rezerva spațiu pentru un fișier pe disc. Acțiunea este întreprinsă de către monitor. Cartela trebuie pusă dacă alocarea de spațiu nu s-a făcut de către S.G.F. (prin cartela LABEL). Forma cartelei este următoarea:

- ALLOC DV:*perif s ... s perif*,FN:*'nume'*[,FS:*f*] [,CD:*dată*][,GN:*g*][,VN:*v*][,RP:*r*][,SZ:*s*]

$$,\text{AM}:\left\{\begin{array}{l}\text{ANY}\\ \text{NEW}\\ \text{SPC}\end{array}\right\}[,\text{VL}:(v_1/s_1[/a_1],\ \ldots,\ v_n/s_n[/a_n])]$$

Informațiile care au apărut în cartelele de comandă ASSIGN, LABEL, INIT și păstrează semnificația din acele cartele.

Argumentul VL este folosit numai dacă opțiunea din AM este SPC. În acest mod avem posibilitatea de a indica cu exactitate aria de pe disc care trebuie alocată fișierului. În argumentul VL apar informațiile următoare:

v_i — identificatorul volumului (format din 6 caractere alfanumerice);

s_i — numărul de cilindri care trebuie alocați ($1 \leqslant s_i \leqslant 199$);

a_i — adresa primului cilindru al acestei zone.

Volumele trebuie să apară în VL în ordinea în care apar în argumentul DV. Aceste volume trebuie să fie premarcate, iar zonele specificate trebuie să fie libere.

Scanned with OKEN Scanner

CARTELA DELETE

Rolul ei este a elibera o zonă de pe disc ocupată de un fișier. Forma ei este următoarea:

• DELETE DV:*perif*,FN:'*nume*'[,VN:*v*][,GN:*g*] [,FP:'*parolă*'][,VP:'*parolă*'][,ON:'*proprietar*'][,UN:*u*]

În cazul cînd dorim eliberarea zonei unei biblioteci (vezi capitolul VII), argumentele GN și VN sînt obligatorii.

Dacă fișierul indicat în cartela DELETE este absent pe disc, rezultatul este tipărirea unui mesaj la consolă, în continuare operatorul fiind cel care decide asupra continuării programului.

Exemplul 1

Considerăm următoarea lucrare:

```
•       JOB BANDA,AN:CCUB,PN:HOPE
•       ** BENZI DE LUCRU PE MT1,MT2
•       COMPILE FORTRAN
* DEFINE FILE SO(DVT:MT,RCF:F,BFS:23)=9
* DEFINE FILE TO(DVT:MT,RCF:F,BFS:23)=8
      INTEGER NUME(5),V,W
      N=0
    2 READ(105,1,END=3)NUME,V
    1 FORMAT(5A4,I3)
      N=N+1
      WRITE(9,1)NUME,V
      GOTO 2
    3 REWIND 9
      I=-1
      M=0
      DO 4 I=1,N
      W=999
      DO 5 J=1,N
      READ(9,1)NUME,V
      IF(V.LT.L.OR.V.EQ.L.AND.J.LE.M)GOTO 5
      IF(W.LE.V)GOTO 5
      K=J
      W=V
```

```
    5 CONTINUE
      REWIND 9
      DO 7 L=1,K
    7 READ(9,1)NUME,V
      WRITE(8,1)NUME,V
      L=V
      M=K
      REWIND 9
    4 CONTINUE
      REWIND 8
      DO 11 I=1,N
      READ(8,1)NUME,V
   11 WRITE(108,12)NUME,V
   12 FORMAT(' ',10X,5A4,4X,I3)
      STOP
      END
.           LINK
.           INIT DV:MT1,VS:111111
.           INIT DV:MT2,VS:222222
.           ASSIGN S,DV:MT1
.           LABEL S,FN:'F1'
.           ASSIGN T,DV:MT2
.           LABEL T,FN:'F2'
.           RUN
 DUMITRU IOANA        167
 VASILESCU ION        187
 RADULESCU SILVIU     183
 IONESCU VASILE       167
 SANDU MIHAI          187
 .EOF
.           EOJ
```

Rezultatele programului vor apare scrise la imprimantă sub forma:

DUMITRU IOANA	167
IONESCU VASILE	167
RĂDULESCU SILVIU	183
VASILESCU ION	187
SANDU MIHAI	187

Scanned with OKEN Scanner

Programul realizează crearea unui fișier pe bandă și apoi, plecînd de la el, crearea unui al doilea fișier pe o altă bandă care să conțină informațiile din primul fișier ordonate conform unui anumit criteriu.

Articolele din cele două fișiere conțin fiecare numele și înălțimea unei persoane. Pentru nume folosim vectorul întreg cu 5 componente NUME, iar pentru înălțime folosim variabila întreagă V.

Prima instrucțiune DEFINE FILE descrie primul fișier. Fișierul are idexul S și va putea fi folosit atît pentru scriere cît și pentru citire (facem însă observația că prin scriere nu înțelegem și modificarea unui articol deja existent în fișier, lucru care nu este posibil). Argumentul DVT arată că fișierul este pe bandă; argumentul RCF arată că folosim formatul fix. Lungimea unui bloc (care coincide cu cea a unui articol) este de 24 octeți (20 pentru nume și 4 pentru înălțime). Numărul de referință este precizat ca fiind 9.

A doua instrucțiune DEFINE FILE descrie al doilea fișier. Informațiile sînt aceleași cu excepția idexului și a numărului de referință, care sînt T și respectiv 8.

Instrucțiunile 4—11 citesc informațiile de pe cartele și înscriu aceste informații în fișierul cu numărul de referință 9. În variabila N se calculează numărul de cartele de date. În momentul în care la citire se întîlnește cartela

- EOF

se execută instrucțiunea 11-a care constă în rebobinarea fișierului la primul său articol.

Instrucțiunile 12—30 ordonează crescător (după înălțime) articolele din fișierul cu idexul S și înscrie articolele ordonate în fișierul cu idexul T. Înainte de executarea pasului I (I = 1, 2,..., N), variabila L va avea ca valoare cea de a I—1 înălțime în ordine crescătoare (inițial L = —1), iar M va memora poziția din fișierul cu indexul S a ultimului articol înscris în fișierul cu indexul T (inițial M = 0). Deci la pasul I ne interesează numai articolele în care înălțimea este cel puțin egală cu L. Dacă există articole în care înălțimea este L, vom determina primul articol care are numărul de ordine mai mare decît M, dîndu-i în același timp ca valoare lui M

numărul de ordine al acestui articol. Dacă nu există articole în care înălțimea este L, vom memora în M numărul de ordine al articolului care conține cea mai mică înălțime mai mare decît L. Apoi fișierul cu idexul S este rebobinat, se citesc primele M articole pentru a-l găsi pe cel cu numărul M și acesta este înscris în fișierul cu idexul T.

Instrucțiunea 31 rebobinează acest ultim fișier.

Instrucțiunile 32—35 permit citirea articolelor din fișierul cu idexul T și listarea lor la imprimantă.

Cartelele de comandă relative la cele două fișiere sînt plasate între cartelele LINK și RUN.

Cartelele INIT inițializează benzile de pe unitățile de bandă MT1 și MT2, înscriind respectiv numerele de serie 111111 și 222222.

Următoarele două cartele de comandă au ca efect atașarea la fișierul de pe unitatea MT1 a idexului S și a identificatorului F1.

Prin ultimele două cartele de comandă referitoare la fișiere, fișierului de pe unitatea de bandă MT2 i se atașează idexul T și identificatorul F2.

Exemplul 2

Considerăm următorul program:

```
.       JOB DISC,AN:CCUB,PN:HOPE
.       COMPILE FORTRAN
* DEFINE FILE *1=1,*2=3
* DEFINE FILE DO(DVT:RD,RCF:D,RCS:24,BFS:24)=7
      INTEGER NUME(5),V,W
      N=0
    2 READ(1,1,END=3)NUME,V
    1 FORMAT(5A4,4X,I3)
      N=N+1
      WRITE(7'N)NUME,V
      GOTO 2
    3 DO 4 I=1,N
      READ(7'I)NUME,V
      W=V
      K=1
```


Scanned with OKEN Scanner

```
      DO 5 J=2,N
      READ(7'J)NUME,V
      IF(W.LE.V)GOTO 5
      W=V
      K=J
5     CONTINUE
      READ(7'K)NUME,V
      V=999
      WRITE(7'K)NUME,V
4     WRITE(7'N+I)NUME,V
      DO 7 I=1,N
      READ(7'N+I)NUME,V
7     WRITE(3,6)NUME,V
6     FORMAT(' ',5A4,3X,I3)
      STOP
      END
```

```
.       LINK
.       INIT DV:RD1,VS:111111
.       ASSIGN D,DV:RD1
.       LABEL D,FN:'FISIER',AM:ANY,SZ:2
.       RUN
IONESCU VASILE          23
VASILESCU ION           18
RADULESCU SILVIU        22
DUMITRU IOANA           19
.EOF
.       EOJ
```

Rezultatele programului vor apare scrise la imprimantă sub forma:

VASILESCU ION 18
DUMITRU IOANA 19
RĂDULESCU SILVIU 22
IONESCU VASILE 23

Scanned with OKEN Scanner

Programul are ca scop crearea unui fișier pe disc și exploatarea lui. Fiecare articol din fișier este constituit din numele și vîrsta unei persoane. Pentru nume folosim vectorul NUME avînd 5 componente întregi, iar pentru vîrstă vom folosi variabilele întregi V și W.

Prin program urmărim ordonarea înregistrărilor din fișier în funcție de vîrstă. Articolele ordonate vor fi scrise în continuarea celor inițiale. Se consideră că informația inițială nu mai este necesară, deci poate fi distrusă.

Prima cartelă DEFINE FILE ne permite să ne adresăm cititorului de cartele și imprimantei și prin numerele de referință 1, respectiv 3 (fără această cartelă ar fi trebuit să folosim obligatoriu numai numerele 105 și respectiv 108).

A doua cartelă DEFINE FILE dă informații despre fișier. Idexul fișierului este D. Fișierul este exploatat și prin scriere și prin citire; este vorba de un fișier pe disc exploatat în acces direct. Lungimea unui articol este de 24 octeți, iar lungimea unui bloc este de 24 octeți. Numărul de referință al fișierului este 7.

Instrucțiunile 4—10 realizează citirea de informații de pe cartele, înscrierea lor în fișier și calcularea în variabila N a numărului de cartele de date, deci de articole înscrise în fișier. În momentul în care prin citire se ajunge la cartela de sfîrșit de date, se trece la executarea celei de a zecea instrucțiuni.

Instrucțiunile 10—21 realizează ordonarea informațiilor din fișier după criteriul vîrstei și înscrierea lor în continuarea articolelor existente. Pentru fiecare I ($1 \leqslant I \leqslant N$) se calculează în W și K cea mai mică vîrstă din cele existente și poziția corespunzătoare. Apoi înregistrarea cu numărul K se modifică înscriind în poziția corespunzătoare vîrstei valoarea 999 pentru a preveni alegerea aceleiași vîrste la pasul următor. Apoi articolul cu numărul K este înscris în fișier pe poziția N+I.

Instrucțiunile 24—26 au ca efect citirea articolelor N+1,..., 2N din fișier (adică articolele inițiale ordonate crescător după vîrstă) și listarea lor la imprimantă.

În continuare descriem cartelele de comandă necesare exploatării fișierului, cartele care se află între LINK și RUN.

Cartela INIT realizează inițializarea discului de pe unitatea de discuri RD1 prin înscrierea pe acest disc a numărului de serie 111111.

Cartela ASSIGN atașează discului de pe unitatea RD1 idexul D. Bineînțeles că acest idex coincide cu cel din ordinul DEFINE FILE.

Cartela LABEL atașează fișierului cu idexul D identificatorul FIȘIER și rezervă pentru acest fișier doi cilindri; alegerea acestor doi cilindri este lăsată la latitudinea sistemului.

Se observă că deși lungimea blocului a fost precizată ca fiind 24 octeți, în realitate un bloc va ocupa 256 octeți, deoarece un bloc ocupă un număr întreg de sectoare.

O folosire mai rațională a discului se poate face precizînd BFS: 240, caz în care un bloc ocupă tot un sector, dar conține 10 articole.

Cea mai economică folosire a discului se realizează precizînd BFS: 768; în acest mod trei sectoare vor conține 32 de articole.

Scanned with OKEN Scanner

Capitolul VII

PROGRAMUL BIBLIOTECAR

DESCRIEREA BIBLIOTECILOR

În esență, o bibliotecă este un fișier folosit exclusiv pentru a păstra programe sursă, în format B.T. sau I.M.T. Existența într-o bibliotecă a unor programe frecvent folosite scutește pe utilizator de munca de a le rescrie, fiind suficient să se facă apelarea lor. De asemenea o bibliotecă permite catalogarea unor programe care sînt folosite periodic, în loc de a păstra cartelele pe care au fost scrise inițial instrucțiunile programului (această ultimă alternativă, pe lîngă faptul că este incomodă, prezintă și dezavantajul deteriorării cartelelor). În plus, așa cum vom vedea în paginile următoare, lucrul cu biblioteci oferă o gamă mai mare de posibilități de rulare a programelor.

Existența bibliotecilor face necesară existența unor programe care să permită crearea și gestionarea eficientă a lor. Această colecție de programe poartă numele de *program bibliotecar* și reprezintă o componentă a sistemului de exploatare.

Programele care fac parte dintr-o bibliotecă poartă numele de *cărți*. O carte poate fi:

— un modul sursă;
— un modul obiect;
— un program obiect.

Bibliotecile pot fi împărțite în două categorii: biblioteci ale sistemului și biblioteci ale utilizatorului.

Bibliotecile sistemului sînt obținute la generarea sistemului. Ele sînt următoarele:

— biblioteca utilizator standard (B.U.S.), care conține programe IMT ale sistemului ca de exemplu compilatoarele, monitorul, editorul de legături, etc.

— biblioteca de subprograme standard ale sistemului (B.S.S.) care conține subprogramele în format B.T. necesare compilatoarelor precum și orice alte subprograme ale utilizatorului de această formă;

— biblioteca de proceduri catalogate (Z%PROC) în care sînt păstrate procedurile catalogate.

Despre ultimele două tipuri de biblioteci vom discuta mai pe larg în capitolele următoare.

Bibliotecile utilizatorului conțin programe ale acestuia, programe într-unul dintre cele trei formate cunoscute. Ele pot fi folosite în fiecare dintre fazele prin care trece un program, după cum urmează:

— la compilare: o carte dintr-o bibliotecă sursă poate fi compilată; modulul obiect rezultat prin compilare poate fi catalogat într-o bibliotecă B.T. (prin specificarea unor opțiuni în cartela COMPILE).

— la editarea legăturilor: modulele obiect dintr-o bibliotecă B.T. pot fi trecute în fișierul de intrare al editorului de legături pentru a fi editate împreună cu alte module; rezultatul editării poate fi catalogat într-o bibliotecă I.M.T. (prin specificarea unor opțiuni în cartela LINK).

— la execuție: programul obiect care trebuie executat poate fi încărcat direct dintr-o bibliotecă I.M.T.

Acele subprograme care sînt utilizate frecvent într-un centru de calcul — subrutinele științifice pentru rezolvarea sistemelor de ecuații liniare, pentru rezolvarea problemelor de programare liniară, pentru generarea de numere aleatoare, etc. — sînt catalogate într-o *bibliotecă de subprograme translatabile*. Cărțile acestei biblioteci sînt în format B.T. Editorul de legături are acces la această bibliotecă și o folosește pentru a include acele module la al căror nume s-a făcut o referire în modulele rezultate prin compilarea programelor utilizatorului.

În lucrul cu biblioteci, cel mai frecvent sînt folosite ca suport discurile magnetice, dar pot fi utilizate și benzile magnetice. Discurile magnetice au avantajele unei capacități mai mari, al accesului mai rapid la informație; în plus, unele

Scanned with OKEN Scanner

funcții ale bibliotecarului, ca de exemplu modificarea instrucțiunilor dintr-un program sursă, nu pot fi executate decît dacă suportul bibliotecii este un disc magnetic.

BIBLIOTECARUL

Programul bibliotecar este, așa cum am văzut, acel program al sistemului de exploatare care se ocupă cu gestionarea cărților din biblioteci. Dintre funcțiile sale amintim următoarele:

— ținerea unei evidențe a cărților din fiecare bibliotecă;
— folosirea cărților din biblioteci în timpul diferitelor faze de rulare a unui program;
— crearea bibliotecilor;
— adăugarea de noi cărți într-o bibliotecă;
— compararea a două biblioteci;
— efectuarea de modificări într-o carte dintr-o bibliotecă sursă;
— înlocuirea sau eliminarea unor cărți din bibliotecă.

Asupra funcțiilor bibliotecarului se va reveni în momentul prezentării cartelelor de comandă ale bibliotecarului prin care se realizează executarea lor.

Apelarea programului bibliotecar se face de către monitor prin introducerea în program a uneia dintre următoarele două cartele de comandă echivalente:

```
•  SYSRUN BIBLIO
•  RUN FN:BIBLIO
```

O bibliotecă este de fapt un fișier format la rîndul său din fișiere. De aceea spunem că o carte este un *fișier membru al unei zone partajate*, care reprezintă de fapt biblioteca.

Cartelele de comandă ASSIGN și LABEL nu sînt aplicabile la lucrul cu biblioteci. Aceasta se datorește faptului că accesul la bibliotecă nu se poate face din cadrul unui program scris în FORTRAN, sau într-un alt limbaj de programare. Informațiile din aceste două cartele de comandă necesare lucrului cu fișiere sînt precizate la deschiderea bibliotecii prin intermediul cartelelor de comandă ale bibliotecarului.

Scanned with OKEN Scanner

Fiecare bibliotecă are atașat un nume. În decursul timpului, bibliotecilor li se aduc însă modificări. De multe ori modificările nu se fac direct în biblioteca respectivă ci se creează o nouă bibliotecă în care cărțile au forma actualizată; păstrarea vechii forme se face pentru acei utilizatori care sînt obișnuiți cu informațiile din ea și nu doresc să schimbe programele pentru a fi compatibile cu noua bibliotecă. Dacă dăm vechiul nume și noii biblioteci, acest nume nu mai este suficient pentru identificarea ei. În capitolul precedent am văzut că fiecărui fișier i se atașează (explicit sau implicit) un număr de generație și un număr de versiune. Același lucru este valabil și pentru biblioteci.

Numărul de generație permite să facem distincție între mai multe biblioteci cu același nume.

Două biblioteci pot avea același nume și același număr de generație. Identificarea lor este posibilă prin intermediul numărului de versiune.

Fiecare carte din bibliotecă are de asemenea un nume. La fel ca pentru biblioteci, o carte, care reprezintă un program al utilizatorului, poate avea mai multe versiuni în timp. Dacă le-am atașa nume distincte, nu ar mai fi evidentă legătura dintre ele. De aceea diferite versiuni ale unei cărți pot fi păstrate în bibliotecă sub același nume, fiecare dintre versiuni avînd în schimb un *număr de actualizare* care permite diferențierea lor. Numărul de actualizare este un număr natural cuprins între 1 și 255. Rezultă deci că o carte din bibliotecă este identificată pe baza numelui său și pe baza numărului de actualizare atașat.

Specificarea în fiecare cartelă de comandă a numelui, numărului de generație și numărului de versiune al unei biblioteci este incomodă. De aceea în cadrul unei lucrări, unei biblioteci i se atașează un identificator de exploatare (idexul bibliotecii), care constă dintr-un caracter alfabetic. Prin aceasta, simpla specificare a acestui caracter într-o cartelă de comandă a bibliotecarului arată că se lucrează cu biblioteca respectivă.

Scanned with OKEN Scanner

GENERALITĂȚI ASUPRA CARTELELOR DE COMANDĂ ALE BIBLIOTECARULUI

Cartelele de comandă ale bibliotecarului au următoarea formă generală:

% *ident listă de argumente* [*comentarii*]

În prima coloană trebuie să apară caracterul %, iar în a doua coloană trebuie să apară un blanc.

ident reprezintă tipul operației; începe din coloana 3 și este urmat de un blanc.

listă de argumente conține argumente despărțite între ele prin virgulă.

Informația de pe o cartelă nu poate depăși coloana 71. Apariția unui asterisc în coloana 72 indică faptul că următoarea cartelă este o cartelă de continuare. Într-o cartelă de continuare în prima coloană trebuie să apară caracterul %, iar în coloanele 2—10 trebuie să apară blancuri; continuarea scrierii argumentelor sau comentariilor se face începînd din coloana 11. În lista de argumente este interzisă apariția unui blanc, cu excepția cazului cînd acest blanc face parte dintr-un șir de caractere încadrat între apostrofuri. Un argument nu poate fi împărțit în două părți care să apară pe cartele diferite; de aceea în asemenea situații, o virgulă de pe o cartelă poate fi urmată pînă la sfîrșitul cartelei de blancuri (următoarea cartelă fiind, bineînțeles, o cartelă de continuare).

Cartelele de comandă ale bibliotecarului trebuie plasate imediat după una dintre cartelele de comandă prezentate mai sus prin care este apelat programul bibliotecar.

Bibliotecarul funcționează sub controlul monitorului. În cursul prelucrării, bibliotecarul face apel la segmente ale monitorului central prin intermediul căruia încarcă în memorie segmentele de care are nevoie. Controlul este redat monitorului la sfîrșitul prelucrării, adică în momentul în care este citită cartela:

% ENDLIB

Bibliotecarul asigură două clase de operații:

— prelucrări de ansamblu asupra bibliotecilor: duplicarea unei biblioteci, reactualizarea unei biblioteci, crearea

unei biblioteci, compararea unor biblioteci sau obținerea repertoriului unei biblioteci.

— prelucrări asupra cărților dintr-o bibliotecă: adăugarea, modificarea sau ștergerea unei cărți, imprimarea sau perforarea unei cărți.

Cartelelor de comandă ale bibliotecarului li se atașează cîte un nivel care este un număr natural cuprins între 1 și 4, după cum urmează:

— *nivelul 1*: OPNLIB
— *nivelul 2*: INCLUDE, REPLACE, DELETE, COPY, DISPLAY, EDIT, REPERT, COMPARE
— *nivelul 3*: RENAME, MOD
— *nivelul 4*: ENDLIB

Cartelele trebuie să apară în ordinea crescătoare a nivelelor atașate. În cadrul aceluiași nivel, cartelele pot apare în general în orice ordine.

O cartelă de tip special numită OUTPUT permite să alegem între a lista sau nu la imprimantă cartelele de date pentru programul bibliotecar. Ea trebuie să apară numai înainte de cartela de nivel 1 sau de cartelele de nivel 2.

Lucrul cu o bibliotecă presupune, ca la orice fișier, deschiderea ei. Aceasta se realizează cu cartela OPNLIB, care trebuie să apară deci în mod obligatoriu. De asemenea este obligatorie apariția cartelei ENDLIB, care indică bibliotecarului sfîrșitul operațiilor pe care le are de executat.

DESCRIEREA CARTELELOR DE COMANDĂ ALE BIBLIOTECARULUI

1) — *Cartela OPNLIB*

Această cartelă are ca efect deschiderea unei biblioteci și stabilirea corespondenței între bibliotecă și idexul ei. Forma ei este următoarea:

% OPNLIB *a*, LN: *nume*, DV: *perif*, VN: *v*, { CD: *dată* / GN: *g* }
,FT: *tip* [,INIT] [,FS: *f*]

Argumentele DV, VN,CD,GN,FS au aceeași formă și semnificație ca pentru orice fișier. Celelalte argumente au următoarele semnificații:

a este idexul bibliotecii; În programe diferite aceeași bibliotecă poate fi utilizată cu idexuri diferite;

nume este numele bibliotecii; este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic;

tip poate avea una din formele:

SOU pentru biblioteci sursă
RBN pentru bibliotecile B.T.
IMT pentru bibliotecile I.M.T.
RSL pentru bibliotecile de subprograme translatabile.

Argumentul INIT trebuie precizat doar la prima referire la bibliotecă.

2) — *Cartela OUTPUT*

Cu ajutorul acestei cartele putem obține la cerere listarea cartelelor de date pentru programul bibliotecar. Forma ei este următoarea:

% OUTPUT $\left\{\begin{matrix} LIST \\ NLST \end{matrix}\right\}$

O opțiune stabilită prin cartela OUTPUT rămîne valabilă pînă la următoarea cartelă OUTPUT indicînd o acțiune contrară. Opțiunea LIST nu se extinde și asupra lucrărilor următoare.

3) – *Cartela ENDLIB*

Această cartelă marchează sfîrșitul cartelelor de comandă ale bibliotecarului. Forma ei este:

% ENDLIB

4) — *Cartela INCLUDE*

Folosirea acestei cartele permite inserarea unei cărți (aflate în fișierul de lucrări sau într-o altă bibliotecă) într-o bibliotecă. Forma ei este următoarea:

% INCLUDE,OL:*b*,OF:*carte*,DL:*a*[,ON:*u*]

Argumentele au următoarele semnificații:
b este idexul fișierului de lucrări (*1) sau al bibliotecii în care se află cartea care trebuie inserată;
carte este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic; reprezintă numele cărții;
a este idexul bibliotecii în care se face catalogarea.
u reprezintă numărul de actualizare ($1 \leqslant u \leqslant 254$).

În cazul în care cartea care trebuie catalogată se află în fișierul de lucrări, ca trebuie să apară imediat după cartela INCLUDE.

Nu este posibilă catalogarea unei cărți I.M.T. care se află în fișierul de lucrări într-o bibliotecă al cărei suport este o bandă magnetică.

5) — *Cartela DELETE*

Această cartelă permite suprimarea unei cărți dintr-o bibliotecă. Forma ei este următoarea:

% DELETE DL:*a*,DF:*carte*[,DN:*u*[,CLR]]

Argumentele au următoarele semnificații:
a este idexul bibliotecii în care se află cartea;
carte este numele cărții și este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic;
u este numărul de actualizare al cărții.

Argumentul CLR are ca efect (atunci cînd este prezent) ștergerea tuturor cărților cu numele *ident* și al căror număr de actualizare este mai mic sau egal cu *u*.

6) — *Cartela REPLACE*

Această cartelă are ca efect înlocuirea unei cărți dintr-o bibliotecă cu alta care se găsește fie într-o bibliotecă, fie în fișierul de lucrări. Forma ei este următoarea:

% REPLACE DL:a_1, DF:$carte_1$[, DN:u_1], OL:a_2[, OF:$carte_2$] [,ON:u_2]

Informațiile a_1, $carte_1$, u_1 identifică cartea care urmează să fie înlocuită.

a_2 este idexul bibliotecii în care se află noua carte sau al fișierului de lucrări.

carte$_2$ este numele cărții care este catalogată în locul cărții vechi, iau a_2 este numărul de actualizare corespunzător. Dacă a_2 nu este specificat, bibliotecarul alege cartea cu cel mai mare număr de actualizare care are numele *carte*$_2$ în biblioteca a_2.

Dacă $a_2 = *1$, cartelele care alcătuiesc cartea care urmează a fi catalogată trebuie să urmeze imediat după cartela REPLACE (sau după cartela RENAME, care va fi descrisă mai jos) în fișierul de lucrări.

Dacă a_2 este idexul unei biblioteci utilizator și cele două cărți au același nume, argumentul OF poate fi omis. În acest caz este însă necesară apariția argumentului DN.

Pentru a putea fi executată acțiunea specificată prin cartela REPLACE, trebuie ca bibliotecile care conțin cele două cărți să fi fost în prealabil deschise.

7) — *Cartela RENAME*

Prin intermediul acestei cartele putem schimba numele unei cărți. Forma ei este următoarea:

% RENAME *carte*

unde *carte* este noul nume al cărții.

Cartela RENAME trebuie să fie precedată fie de cartela REPLACE, fie de cartela INCLUDE. În aceste cartele apare numele cărții care urmează să înlocuiască o altă carte sau să fie inclusă în bibliotecă. Acest nume poate fi schimbat prin plasarea cartelei RENAME imediat după una dintre cartelele amintite.

Dacă în cartelele REPLACE sau INCLUDE apare argumentul OL:*1 și aceste cartele sînt urmate de RENAME, cartelele conținînd cartea care trebuie inclusă în bibliotecă trebuie să urmeze lui RENAME.

De exemplu cartelele:

```
% INCLUDE OL:A,OF:CARTX,ON:3,DL:B
% RENAME CARTY
```

au ca rezultat catalogarea în biblioteca cu idexul A a cărții cu numele CARTX și numărul de actualizare 3 din biblioteca cu idexul B. Această carte va apare în biblioteca cu idexul A sub numele CARTY.

Scanned with OKEN Scanner

Efectul cartelelor următoare:

```
% REPLACE DL:A,DF:CARTE,DN:2,OL:A,ON:2
% RENAME CARTI
```

este schimbarea în CARTI a numelui cărții CARTE care apare în biblioteca A cu numărul de actualizare 2.

8) — *Cartela COPY*

Această cartelă permite copierea unei biblioteci în altă bibliotecă. Forma ei este următoarea:

% COPY DL:a_1,OL:a_2

unde a_1 și a_2 sînt respectiv idexurile bibliotecii în care se copiază și a bibliotecii care este copiată. Biblioteca cu idexul a_1 poate să conțină deja cărți catalogate. Evident, bibliotecile trebuie să fie de același tip și să fi fost în prealabil deschise. Cele două biblioteci se pot afla fie pe bandă, fie pe disc.

9) — *Cartela DISPLAY*

Cu ajutorul acestei cartele putem realiza imprimarea sau perforarea unei cărți, a unui segment sau a unui subprogram dintr-o bibliotecă de subprograme. Forma ei este următoarea:

$$\%\ \text{DISPLAY}\ \text{OL:a,}\ \left\{\begin{array}{l}\text{SP:}carte\\ \\ \text{OF:}carte[,\text{ON:}u][,\text{OS:}nume]\end{array}\right\}\left[,\left\{\begin{array}{l}\text{LIST}\\ \text{PUN}\\ \text{LIST,PUN}\end{array}\right\}\right]$$

unde argumentele au următoarea semnificație:

a este idexul bibliotecii.

carte este sau numele unui subprogram din biblioteca de subprograme sau numele unei cărți dintr-o bibliotecă utilizator. În mod corespunzător, *carte* este format din 1—6 respectiv 1—8 caractere alfanumerice dintre care primul este alfabetic.

nume este numele segmentului care trebuie perforat sau listat.

Argumentul LIST cere listarea la imprimantă, PUN cere perforarea la perforatorul de cartele, iar LIST,PUN implică ambele acțiuni.

10) — *Cartela REPERT*

Această cartelă permite tipărirea unui repertoar alcătuit din numele cărților care fac parte dintr-o bibliotecă. Pentru bibliotecile de subprograme se obțin și numele punctelor de intrare, iar pentru bibliotecile I.M.T. se obțin și numele segmentelor conținute în fiecare carte. Tot în cazul bibliotecilor I.M.T. se poate obține la cerere numai numele segmentelor dintr-o anumită carte:

Forma cartelei este următoarea:

% REPERT OL:*a*[,OF:*ident*[,ON:*u*]]

unde semnificația argumentelor este cea cunoscută. Argumentele OF și ON se folosesc atunci cînd dorim să obținem numai repertoarul segmentelor din care este formată o carte dintr-o bibliotecă I.M.T.; cartea este identificată prin intermediul acestor două argumente.

11) — *Cartela EDIT*

În bibliotecile sursă cărțile reprezintă programe sursă catalogate în vederea unei utilizări ulterioare. De multe ori este necesară modificarea acestor programe, în sensul înlocuirii unor instrucțiuni prin altele, adăugarea unor instrucțiuni noi, etc. Acestă modificări sînt posibile prin folosirea cartelei EDIT, legată și de folosirea cartelei MOD. Forma cartelei este următoarea:

% EDIT OL:*a*,OF:*carte*[,ON:*u*][,ST:*pas*[,IN:*val*]]

$$\left[\left\{\begin{matrix}\text{,DEL}\\ \text{,RET}\end{matrix}\right\}\right]\left[\left\{\begin{matrix}\text{,LIST}\\ \text{,NLST}\end{matrix}\right\}\right]$$

Informațiile *a*, *carte*, *u* identifică cartea care urmează a fi prelucrată.

Argumentele ST și IN servesc la identificarea instrucțiunilor din cartea respectivă.

O carte sursă este formată din imaginile cartelelor care alcătuiesc programul sursă. În bibliotecă aceste cartele sînt numerotate, ele primind un *număr de secvență*, număr care va fi plasat în coloanele 73—80 în cazul programelor scrise în FORTRAN.

Argumentul *val* reprezintă numărul care va fi înscris în prima cartelă a cărții; este format din 1—8 cifre zecimale.

Argumentul *pas* este un număr zecimal format din cel mult 3 cifre, dintre care cel mult una poate fi diferită de zero. Numărul de secvență al unei cartele se obține cu ajutorul formulei *val* + (k—1)**pas* unde k este numărul de ordine al cartelei în program. Acest număr de secvență va fi folosit în cartela MOD pentru a identifica acea cartelă începînd de la care se inserează un șir de cartele, se șterg sau se fac modificări de cartele.

Prin lipsă, *val* = *pas* = 1.

Precizăm că în cartela EDIT argumentele *val* și *pas* se referă la cartea care va lua naștere, iar în cartela MOD aceste valori se referă la cartea care urmează a fi actualizată.

Ca rezultat al acțiunii conjugate a cartelelor EDIT și MOD se obține o nouă carte, care va avea același nume dar un număr de actualizare cu o unitate mai mare. Vechea carte poate fi păstrată sau nu în continuare în bibliotecă, după cum este în vigoare opțiunea RET sau opțiunea DEL.

Cartea actualizată va fi listată la imprimantă sau nu, după cum este specificat argumentul LIST sau argumentul NLST.

O cartelă EDIT trebuie să fie urmată totdeauna de cel puțin o cartelă MOD.

Observație: Dacă argumentul ON: *u* nu apare într-una dintre cartelele DISPLAY, REPERT, EDIT, este considerată cartea cu cel mai mare număr de actualizare dintre cele identificate prin celelalte argumente.

12) — *Cartela MOD*

Cartela MOD permite modificarea conținutului unei cărți dintr-o bibliotecă sursă prin ștergerea, inserarea sau înlocuirea unui șir de cartele din cartea sursă cu cartele din fișierul de lucrări. Cartea sursă care trebuie modificată este identificată prin intermediul cartelei EDIT, care în plus stabilește și numerele de secvență ale cartelelor din cartea care va lua naștere. Forma cartelei MOD este următoarea:

% MOD n_1 [, n_2]

Prin n_1 indicăm numărul de secvență al cartelei după care se inserează un șir de cartele sau al cartelei începînd de la care se face ștergerea sau înlocuirea. Prin n_2 se precizează numărul de secvență al ultimei cartele din șirul de cartele care trebuie șters sau înlocuit.

Numerele n_1 și n_2 sînt formate din 1—8 cifre zecimale.

Dacă n_2 lipsește, efectul cartelei MOD va fi inserarea unui șir de cartele. Cartelele care sînt inserate se află în fișierul de lucrări, imediat după cartela MOD.

Ștergerea unui șir de cartele se face prin precizarea lui n_1 și n_2 în cartela MOD. Dacă $n_1 = n_2$, se șterge o singură cartelă.

Înlocuirea unui șir de cartele cu un altul se face de asemenea dacă în cartela MOD este specificat atît n_1 cît și n_2. Cartelele înlocuitoare se află în fișierul de lucrări imediat după cartela MOD. Dacă $n_1 = n_2$, se va înlocui cartela cu numărul de secvență n_1 cu șirul de cartele aflat în fișierul de lucrări.

Din cele de mai sus rezultă că în cadrul apariției ambelor numere n_1 și n_2 acțiunea întreprinsă este cea de ștergere sau cea de înlocuire după cum după cartela MOD nu apar, respectiv apar, cartele conținînd instrucțiuni sursă.

Exemplul 1

Considerăm următoarea lucrare:

```
*          JOB XXX,AN:COUB,PN:HOTE
*          INIT DV:RD1,VS:111111
*          ALLOC DV:RD1,FN:'ABCDEFGHSOU',AM:ANY,SZ:10
*          SYSRUN BIBLIO
% OPNLIB S,LN:ABCDEFGH,DV:RD1,VN:0,GN:1,FT:SOU,INIT
% OUTPUT LIST
% INCLUDE OL:*1,OF:PROGRAMA,DL:S
      FUNCTION CA(N,K)
      INTEGER CA
      CA=1
      DO 1 I=1,K
    1 CA=CA*(N+1 I)
      RETURN
      END
% DISPLAY OL:S,OF:PROGRAMA
% REPERT OL:S
% ENDLIB
*          COMPILE FORTRAN
      INTEGER CA
      N=10
      K=5
      I=CA(N,K)
      WRITE(102,1)N,K,I
    1 FORMAT(3I8)
      STOP
      END
*          COMPILE FORTRAN
* FETCHS LN:ABCDEFGH,DV:RD1,FN:PROGRAMA
*          LINK
*          RUN
*          EOJ
```

Programul începe cu crearea unei biblioteci sursă. Pentru aceasta se inițializează mai întîi discul de pe unitatea RD1

și se rezervă 10 cilindri pentru biblioteca sursă cu numele ABCDEFGH.

Cartela SYSRUN BIBLIO aduce în memorie programul bibliotecar și lansează execuția lui.

Cartela OPNLIB deschide biblioteca cu numele ABCDEFGH aflată pe unitatea de discuri RD1; biblioteca este în format sursă, are numărul de versiune 0 și numărul de generație 1. Idexul bibliotecii este S, identificator prin intermediul căruia va fi recunoscută în continuare. Este folosit argumentul INIT, deoarece urmează să se facă prima scriere în bibliotecă.

Cartela OUTPUT, în care este folosit argumentul LIST, determină listarea datelor pentru cartelele de comandă ale bibliotecarului, deci în cazul nostru a cartelelor care conțin programul sursă care trebuie catalogat.

Cartela INCLUDE introduce cartea cu numele PROGRAMA în biblioteca cu idexul S. Cartea este constituită din programul care se află în fișierul de lucrări imediat după cartela INCLUDE. Programul inclus este un subprogram funcție care calculează A_N^K.

Cartela DISPLAY listează cartea PROGRAMA din biblioteca cu idexul S. Lista cuprinde numele bibliotecii, numărul de versiune, numărul de generație, tipul bibliotecii și instrucțiunile din cartea respectivă. În cazul nostru vom obține următoarea listă:

```
NOM DE LA BIBLIOTHEQUE :          ABCDEFGH
      NUMERO DE VERSION :         00
   NUMERO DE GENERATION :         0001
                 FORMAT :         SOU
NOM DU FICHIER : PROGRAMA   NUMERO DE MISE A JOUR :   1
    01          FUNCTION CA(N,K)
    02          INTEGER CA
    03          CA=1
    04          DO 1 I=1,K
    05        1 CA=CA*(N+1-I)
    06          RETURN
    07          END
```

Cartela REPERT întocmește repertoarul bibliotecii cu idexul S, listînd numele bibliotecii, numerele de versiune și generație, precum și tipul bibliotecii; în continuare sînt listate numele și numerele de actualizare ale cărților din bibliotecă. În cazul nostru, obținem lista:

```
NOM DE LA BIBLIOTHEQUE :          ABCDEFGH
     NUMERO DE VERSION :          00
  NUMERO DE GENERATION :          0001
                FORMAT :          SOU
NOM DU FICHIER :                  NUMERO DE MISE A JOUR :
                PROGRAMA                                  1
```

Cartela ENDLIB indică sfîrșitul cartelelor de comandă ale bibliotecarului.

În continuare este compilat un program sursă care constă în apelarea subprogramului funcție CA pentru valori ale argumentelor egale cu 10 și 5 și scrierea la imprimantă a rezultatului obținut.

Cartela FETCHS care urmează încarcă în fișierul de lucrări cartea PROGRAMA pentru a fi compilată. Forma generală a acestei cartele va fi prezentată în capitolul 9.

Urmează cartelele care indică editarea legăturilor și executarea programului obiect obținut. Ca rezultat se va obține valoarea $30240 = A_{10}^{5}$.

Exemplul 2

Considerăm următoarea lucrare, care va folosi biblioteca creată prin lucrarea din exemplul 1.

```
*          JOB XYZ,AN:OCUB,PN:HOPE
*          SYSRUN BIBLIO
% OPNLIB S,LN:ABCDEFGH,DV:RD1,FT:SOU,VN:0,GN:1
% EDIT OL:S,OF:PROGRAMA,ST:10,IN:100
% MOD 5
      DO 2 I=1,K
    2 CA=CA/I
% DISPLAY OL:S,OF:PROGRAMA
% ENDLIB
```

```
.          COMPILE FORTRAN
* FETCH: LN:ABCDEFGH,DV:RD1,FN:PROGRAMA
          COMPILE FORTRAN
          INTEGER CA
          N=10
          K=5
          I=CA(N,K)
          WRITE(108,1)N,K,I
        1 FORMAT(3I8)
          STOP
          END
.          LINK
.          RUN
.          EOJ
```

Cartela SYSRUN BIBLIO lansează programul bibliotecar. Cartela OPNLIB deschide biblioteca ABCDEFGH, atașîndu-i idexul S.

Cartelele EDIT și MOD care urmează creează o nouă carte în biblioteca cu idexul S. Noua carte este obținută prin inserarea a două instrucțiuni aflate în fișierul de lucrări în cartea cu numele PROGRAMA aflată în biblioteca cu idexul S. Noua carte va avea tot numele PROGRAMA, dar numărul de actualizare atașat va fi cu o unitate mai mare decît al cărții plecînd de la care s-au făcut modificările, deci va fi egal cu 2. În noua carte obținută, numerele de secvență atașate instrucțiunilor vor forma o progresie aritmetică cu primul termen egal cu 100 și cu rația 10.

Cartela DISPLAY cere listarea cărții cu numele PROGRAMA aflată în biblioteca cu idexul S. Cum nu este specificat nici un număr de actualizare, va fi listată cartea cu numele dat și avînd cel mai mare număr de actualizare din biblioteca respectivă, deci cartea cu numărul de actualizare 2 care tocmai a fost creată în fișier (facem observația că și cartea cu numele PROGRAMA și numărul de actualizare 1 a rămas în bibliotecă, deoarece prin lipsă este valabil argumentul RET din cartela EDIT).

La imprimantă se va obține lista:

```
NOM DE LA BIBLIOTHEQUE :          ABCDEFGH
     NUMERO DE VERSION :          00
  NUMERO DE GENERATION :          0001
                FORMAT :          SOU
NOM DU FICHIER : PROGRAMA     NUMERO DE MISE A JOUR : 2
    01          FUNCTION CA(N,K)
    02          INTEGER CA
    03          CA=1
    04          DO 1 I=1,K
    05        1 CA=CA*(N+1-I)
    06          DO 2 I=1,K
    07        2 CA=CA/I
    08          RETURN
    09          END
```

Cartela ENDLIB delimitează sfîrșitul cartelelor de comandă ale bibliotecarului.

În continuare sînt compilate un program principal, identic cu cel din exemplul precedent, precum și (cu ajutorul cartelei FETCHS) cartea cu numele PROGRAMA avînd cel mai mare număr de actualizare dintre cărțile cu acest nume din biblioteca cu numele ABCDEFGH.

Urmează editarea legăturilor și lansarea programului obiect obținut. Ca rezultat se va obține valoarea $252 = C_{10}^5$.

Scanned with OKEN Scanner

Capitolul VIII

SEGMENTAREA PROGRAMELOR

GENERALITĂȚI DESPRE SEGMENTARE

În urma compilării unui program sursă FORTRAN cu ajutorul unei cartele COMPILE avînd forma prezentată în capitolul 3, obținem ca rezultat:

— un modul obiect program;

— atîtea module de date comune cîte blocuri de date comune există în program.

Modulul obiect program conține instrucțiunile în limbaj mașină rezultate în urma compilării, adresele operanzilor relative la adresa de început a modulului, informațiile necesare pentru a face legătura cu alte module la care s-a făcut referire în program, datele (constante, variabile) care apar în program (cu excepția datelor care apar în instrucțiunile COMMON).

Un modul de date comune conține toate variabilele declarate prin COMMON ca făcînd parte din același bloc de date comune.

În toate programele cu care am lucrat pînă acum, editorul de legături stabilea legăturile între modulele obiect, iar programul obiect obținut era încărcat în memorie pentru a fi lansată execuția lui. Deci în această situație, atît programul principal cît și toate subprogramele folosite apar simultan în memorie.

Una dintre problemele care apare foarte frecvent în programare este cea a capacității limitate a memoriei interne. Această limitare face ca pentru programe complexe, de dimensiuni mari, care folosesc un număr mare de subprograme și un volum mare de date, memoria internă a calculatorului să nu fie suficientă pentru păstrarea programelor și datelor,

ceea ce face imposibilă rularea programului în forma prezentată în capitolele anterioare.

În aceste condiții, pentru a fi totuși posibilă prelucrarea cerută, a fost introdusă o tehnică specială, numită *segmentare*. Ideea de bază a segmentării este aceea de a nu păstra în memorie decît unitățile de program de care este nevoie la fiecare moment de timp. Principala justificare pentru folosirea acestei tehnici rezidă în faptul că uneori, în cadrul aceleiași lucrări, un subprogram este apelat la un anumit moment de timp și apoi nu mai este folosit pînă la sfîrșit sau pînă la următoarea sa apelare, ocupînd astfel inutil memoria.

Segmentarea permite aducerea în memorie a unor părți ale lucrării doar atunci cînd este nevoie de ele, încărcarea în memorie făcîndu-se peste alte unități de program care nu mai sînt necesare în acel moment al execuției. Utilizatorul are posibilitatea să indice modul în care, prin încărcare, unitățile de program se suprapun; pentru aceasta el trebuie să precizeze o structură de acoperire prin cartela de comandă TREE.

În general, nu toate unitățile de program folosesc toate blocurile de comun. Segmentarea permite și încărcarea în memorie a unui bloc sau a mai multe blocuri COMMON doar în momentul în care informația din ele trebuie folosită, încărcarea făcîndu-se de asemenea într-o zonă a memoriei devenită disponibilă în acel moment al execuției.

În cadrul segmentării, unitatea de bază din punctul de vedere al încărcării în memoria internă este *segmentul*. Segmentele iau naștere în urma editării legăturilor, dar pregătirea formării lor se face încă din faza de compilare. În sistemul SIRIS 3 distingem trei tipuri de segmente și anume:

— segmente de program;
— segmente de date;
— segmente de date comune.

După cum am mai spus, în mod normal compilatorul FORTRAN crează un singur modul pentru instrucțiunile și datele unei unități de program. Este însă posibil să creăm

un modul distinct pentru date, rămînînd un segment program pur; de asemenea la compilare putem indica blocurile de comun care vor face parte din același segment de date comune.

Un segment de program este format din module obiect ale utilizatorului, module obiect care reprezintă rutine ale sistemului și funcții in-line (radical, modul, funcții trigonometrice, etc.). Orice segment de program capătă un nume prin intermediul cartelei de comandă SEG.

Un segment de date este format din module de date. Segmentele de date sînt plasate în continuarea segmentelor de program corespunzătoare; lor nu li se atașează un nume distinct de cel al segmentelor de program, deoarece sînt indicate implicit de acestea și încărcate în memorie împreună cu acestea.

Un segment de date comune este format din module de date comune. Un astfel de segment capătă automat ca nume numele primului modul de date comune care intră în compunerea sa (deci nu poate fi precizat de utilizator). Segmentele de date comune sînt încărcate în secvență cu segmentele de program, conform informațiilor din cartela TREE.

Relațiile dintre segmente pot fi descrise grafic printr-un *arbore*, care corespunde structurii de acoperire a segmentelor programului. Arborele precizează pentru fiecare segment legăturile lui cu alte segmente precum și ce segmente acoperă prin încărcarea lui în memorie.

Segmentul de la baza arborelui se numește *rădăcină* și rămîne în memorie pe toată durata executării programului. Programul principal trebuie să facă obligatoriu parte din segmentul rădăcină.

ORDINELE SEGMENT ȘI SEGDATA

În FORTRAN, pentru pregătirea formării segmentelor de date și de date comune se folosesc ordinele SEGMENT și SEGDATA. Aceste două ordine trebuie plasate în cadrul

lucrării după cartela COMPILE, dar înaintea primei instrucțiuni a unității de program respective. Pentru fiecare unitate de program putem folosi cel mult unul dintre aceste ordine.

1) — *Ordinul* SEGMENT

Fiecare bloc de date comune va da naștere în urma compilării la un modul de date comune, care are același nume ca și blocul de date comune. Prin ordinul SEGMENT indicăm care sînt acele module de date comune pe care dorim să le includem într-un același segment. Forma ordinului este următoarea:

* SEGMENT $[s_1[,s_2[,s_3[,s_4]]]]$

Sintaxa cartelei SEGMENT (ca și cea a cartelei SEGDATA) este aceeași cu cea a ordinului DEFINE FILE, care a fost descris anterior.

Fiecare dintre argumentele s_i (i = 1, 2, 3, 4) poate avea una din formele:

$$c$$
$$c_1 + c_2 + \ldots + c_p$$

unde c, respectiv c_k sînt numele FORTRAN ale blocurilor de comun etichetate sau identificatorul F%BLK, care corespunde blocului de comun neetichetat.

Fiecare argument s_i definește blocurile de comun care vor face parte din același segment de date comune.

În cazul în care în ordinul SEGMENT nu apar segmente, toate blocurile de comun (etichetate sau nu) alcătuiesc un singur segment de date comune.

2) — *Ordinul* SEGDATA

Forma acestui ordin este următoarea:

* SEGDATA [*(nume)*][s_1 [,s_2 [,s_3]]]

Ordinul SEGDATA are același efect ca și ordinul SEGMENT, formînd în plus un modul de date cu datele locale ale unității de program respective.

Scanned with OKEN Scanner

nume este un identificator atașat modulului de date care se formează; este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic. Acest nume trebuie precizat numai pentru subprograme, deoarece pentru programul principal modulul de date care ia naștere capătă automat numele F%MDATA; *nume* trebuie să fie diferit de numele blocurilor de comun, numele subprogramelor și numele de intrare care apar în program; de asemenea numele diferitelor module de date trebuie să fie diferite între ele.

CARTELE DE COMANDĂ FOLOSITE LA SEGMENTARE

Există două astfel de cartele și anume cartelele SEG și TREE. Ele definesc segmentele și structura arborelui care descrie relațiile între segmente.

1) — *Cartela SEG*

Această cartelă de comandă servește la constituirea și identificarea unui segment de program și eventual a unui segment de date (dacă a fost folosit ordinul SEGDATA). Forma ei este următoarea:

- SEG *ident*

unde *ident* este numele segmentului de program și este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic.

Segmentul program va fi format din toate modulele obiect program care urmează după această cartelă și pînă la următoarea cartelă SEG sau pînă la cartela LINK împreună cu modulele sistemului și funcțiile in-line necesare. Segmentul de date va fi format din toate modulele de date care au fost create separat de modulele obiect program precizate mai sus.

2) — *Cartela TREE*

Această cartelă precizează structura arborelui prin descrierea modului în care segmentele se suprapun la încărcarea

lor în memorie. Forma ei este următoarea:

TREE *arbore*

unde *arbore* este o expresie în care intervin nume de segmente și operatori. După cum am văzut, numele de segmente sînt formate din 1—8 caractere alfanumerice, dintre care primul este alfabetic. Pentru segmentele de date comune folosim sau numele unui bloc cuprins între paranteze sau o expresie între paranteze în care apar (legate prin +) modulele de date comune grupate în segmentul de date comune respectiv.

Operatorii sînt în număr de patru și au următoarele semnificații:

(— segmentele care urmează sînt de nivel inferior

) — sfîrșitul expresiei sau urmează un segment de nivel superior segmentului sau segmentelor care au apărut între această paranteză închisă și paranteza deschisă corespunzătoare.

, — segmentul care urmează este de același nivel ca segmentul de la stînga virgulei, deci se va suprapune peste el în momentul încărcării

\+ — segmentul care urmează este un segment de date comune atașat segmentului program precedent

Segmentul rădăcină (care este unic) este primul segment care apare în arbore.

Cartela TREE trebuie să apară imediat înainte de cartela LINK. În lipsa cartelei TREE, programul are structură liniară.

Observații

1) — La același nivel pot exista mai multe segmente de date comune.

2) — Informațiile din ordincle SEGMENT și SEGDATA trebuie să fie compatibile cu cele din cartela TREE.

3) — Încărcarea în memorie a unui segment de program (deci implicit și a eventualului segment de date și a eventualelor segmente de date comune atașate) se face în momentul în care în timpul execuției programului se întîlnește o

referire la numele unuia dintre subprogramele care intră în alcătuirea segmentului.

Încărcarea se face în continuarea segmentului de nivel imediat superior. Dacă un segment se află în memorie, atunci toate segmentele care apar pe ramura care îl leagă de segmentul rădăcină se află de asemenea în memorie.

4) — Este sarcina programatorului ca, prin cartela TREE, să determine încărcarea la timp a segmentelor de comun și ca în orice moment al execuției să asigure prezența în memorie a subprogramelor care sînt apelate.

5) — Facilitatea de a forma segmente de date a fost introdusă pentru perfecționări ulterioare ale sistemului. Este însă efectiv necesară folosirea ei în situația în care un modul obiect necesită o memorie mai mare decît 64 Kocteți (lucru nepermis de sistem); în acest caz, prin crearea unui modul separat de date putem obține două module necesitînd fiecare mai puțin de 64 Kocteți.

Scanned with OKEN Scanner

Exemplu:

Considerăm următoarea lucrare:

```
.           JOB CCC,AN:CCUB,PN:HOPE
.           SEG PRINC
.           COMPILE FORTRAN
* SEGMENT F%BLK+ALFA
      IMPLICIT INTEGER(A-Z)
      COMMON V(12)/ALFA/PAR,IMPAR
      READ(105,1)V
    1 FORMAT(12I3)
      WRITE(108,2)V
    2 FORMAT('1',10X,'VECTORUL V'/' ',12I3//)
      CALL TSTPAR(12)
      WRITE(108,3)PAR,IMPAR
    3 FORMAT(' ',5X,'IN VECTOR SINT'/
     +' ',10X,I3' NUMERE PARE'/
     +' ',6X,'SI',I3,' NUMERE IMPARE'//)
      CALL ORDVCT(12)
      WRITE(108,4)V
    4 FORMAT(' ',10X,'** VECTORUL ORDONAT **'/
     +' ',12I3//)
      CALL MATOP
      WRITE(108,5)
    5 FORMAT(7X,'**** SFIRSIT ****')
      STOP
      END
.           SEG PAR
.           COMPILE FORTRAN
* SEGMENT F%BLK+ALFA
      SUBROUTINE TSTPAR(N)
      IMPLICIT INTEGER(A-Z)
      COMMON V(12)/ALFA/PAR,IMPAR
      PAR=0
      IMPAR=0
```

Scanned with OKEN Scanner

```
      DO 3 K=1,N
      IF(MOD(V(K,2)))1,2,1
    1 IMPAR=IMPAR+1
      GOTO 3
    2 PAR=PAR+1
    3 CONTINUE
      RETURN
      END
          SEG MAXIM
          COMPILE FORTRAN
SEGDATA(DATAMAX)F%BLK
      SUBROUTINE MAX(N,POZ)
      IMPLICIT INTEGER(A-Z)
      COMMON X(12)
      POZ=1
      DO 1 K=2,N
    1 IF(X(POZ).LE.X(K))POZ=K
      RETURN
      END
          SEG OVCT
          COMPILE FORTRAN,MAP,OBL
SEGMENT F%BLK
      SUBROUTINE ORDVCT(N)
      IMPLICIT INTEGER(A-Z)
      COMMON X(12)
      K=N
    1 CALL MAX(K,IND)
      AUX=X(IND)
      X(IND)=X(K)
      X(K)=AUX
      K=K-1
      IF(K.GT.1)GOTO 1
      RETURN
      END
```

Scanned with OKEN Scanner

```
.          SEG A
.          COMPILE FORTRAN
* SEGMENT F%BLK,BETA
      SUBROUTINE MATOP
      IMPLICIT INTEGER(A-Z)
      COMMON X(12)/BETA/A(3,4),B(3,4)
      READ(105,1)A,B
    1 FORMAT(12I3)
      WRITE(108,2)(((A(I,J),J=1,4),
     +(B(I,J),J=1,4)),I=1,3)
    2 FORMAT(5X,'MATRICEA A          MATRICEA B'//
     +(4X,4I3,4X,4I3)//)
      CALL SUMMAT
      CALL MAXMAT(VAL)
      WRITE(108,3)VAL
    3 FORMAT(13X,'ELEMENTUL MAXIM AL SUMEI ESTE',I4)
      RETURN
      END
.          SEG B
.          COMPILE FORTRAN
* SEGMENT F%BLK,BETA
      SUBROUTINE SUMMAT
      IMPLICIT INTEGER(A-Z)
      COMMON X(12)/BETA/A(3,4),B(3,4)
      DIMENSION SUM(3,4)
      EQUIVALENCE(X(1),SUM(1,1))
      DO 1 J=1,4
      DO 1 I=1,3
    1 SUM(I,J)=A(I,J)+B(I,J)
      WRITE(108,2)((SUM(I,J),J=1,4),I=1,3)
    2 FORMAT(11X,'MATRICEA SUMA'//(12X,4I3)//)
      RETURN
      END
```

Scanned with OKEN Scanner

```
.          SEG C
.          COMPILE FORTRAN
* SEGMENT F%BLK,BETA
       SUBROUTINE MAXMAT(VAL)
       IMPLICIT INTEGER(A-Z)
       COMMON X(12)
       CALL MAX(12,POZ)
       VAL=X(POZ)
       RETURN
       END
       TREE PRINC+(F%BLK+ALFA)(PAR,MAXIM(OBCT,A+(BETA)(B,C)))
.      LINK
.      RUN
.
 12  2  5  1  8 16 21 13  9  5 18  0
 13 20 31  8  2 31 19 16 19 14 10  8
 24 23  2 13 19  9  7 15 43 21 20  5
.EOF
.          EOJ
```

În urma execuției programului, obținem rezultatele:

```
          VECTORUL V
12  2  5  1  8 16 21 13  9  5 18  0

    IN VECTOR SINT
          6 NUMERE PARE
      SI 6 NUMERE IMPARE
          ** VECTORUL ORDONAT **
 0  1  2  5  5  8  9 12 13 16 18 21

    MATRICEA A          MATRICEA B
    13  8 19 14          24 13  7 21
    20  2 16 10          23 19 15 20
    31 31 19  8           2  9 43  5

          MATRICEA SUMA
             37 21 26 35
             43 21 31 30
             33 40 62 13
          ELEMENTUL MAXIM AL SUMEI ESTE     62
*  **** SFIRSIT ****
```


9 — Introducere în sistemul de operare Siris

```
.          SEG C
.          COMPILE FORTRAN
* SEGMENT F%BLK,BETA
      SUBROUTINE MAXMAT(VAL)
      IMPLICIT INTEGER(A-Z)
      COMMON X(12)
      CALL MAX(12,POZ)
      VAL=X(POZ)
      RETURN
      END
      TREE PRINC+(F%BLK+ALFA)(PAR,MAXIM(OBCT,A+(BETA)(B,C)))
.     LINK
.     RUN
.
 12  2  5  1  8 16 21 13  9  5 18  0
 13 20 31  8  2 31 19 16 19 14 10  8
 24 23  2 13 19  9  7 15 43 21 20  5
.EOF
.          EOJ
```

În urma execuției programului, obținem rezultatele:

```
        VECTORUL V
12  2  5  1  8 16 21 13  9  5 18  0

   IN VECTOR SINT
         6 NUMERE PARE
    SI 6 NUMERE IMPARE
       ** VECTORUL ORDONAT **
 0  1  2  5  5  8  9 12 13 16 18 21

  MATRICEA A          MATRICEA B
   13  8 19 14          24 13  7 21
   20  2 16 10          23 19 15 20
   31 31 19  8           2  9 43  5

        MATRICEA SUMA
           37 21 26 35
           43 21 31 30
           33 40 62 13
        ELEMENTUL MAXIM AL SUMEI ESTE     62
**** SFIRSIT ****
```

Scanned with OKEN Scanner

Înainte de a trece la descrierea lucrării prezentate, facem observația că ea reprezintă un exemplu pur didactic, deoarece unitățile de program care alcătuiesc lucrarea necesită o cantitate de memorie mică, astfel încît nu era necesar să folosim segmentarea, fiind posibilă încărcarea în memorie a tuturor modulelor obiect obținute.

Prelucrările efectuate de program sînt următoarele:

— este citit un vector cu 12 componente întregi și apoi se determină numărul de elemente pare și impare din acest vector; pentru aceasta se folosește subrutina TSTPAR;

— este apelată subrutina ORDVCT care ordonează crescător elementele vectorului; pentru aceasta este folosită subrutina MAX, care găsește poziția POZ a elementului maxim din vectorul V de lungime N, declarat în zonă comună;

— este apelată subrutina MATOP, care citește de pe cartele două matrici A și B și apelează subrutina SUMMAT care calculează suma celor două matrici; rezultatul este listat la imprimantă. Tot din subrutina MATOP este apelată subrutina MAXMAT, care, cu ajutorul subrutinei MAX, calculează elementul maxim al matricei sumă obținute anterior.

Facem următoarele observații:

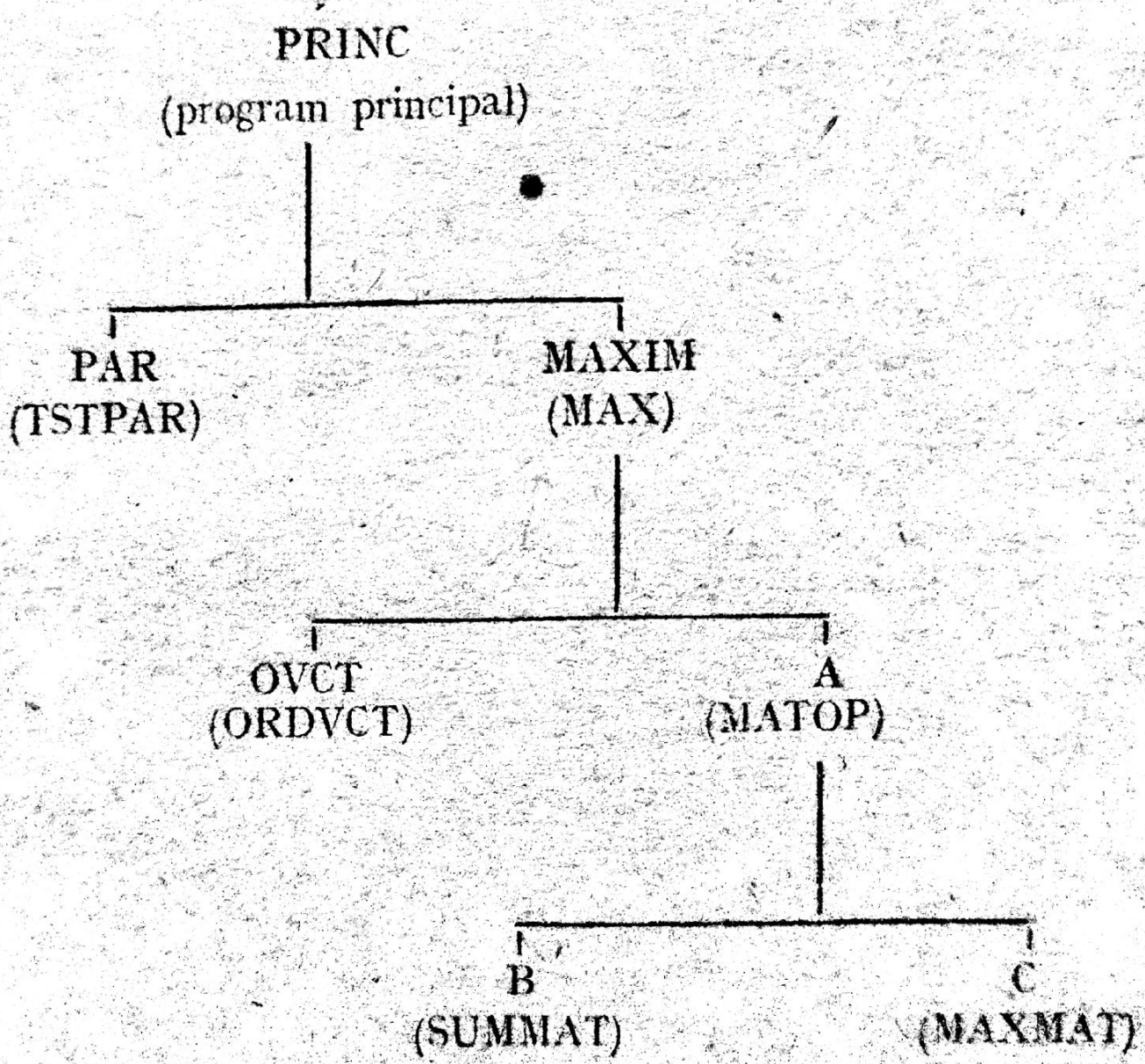
— după ce a fost apelată din programul principal, subrutina TSTPAR nu mai este folosită;

— subrutina MAX nu este folosită de subrutina TSTPAR, dar este folosită de subrutinele ORDVCT și MATOP (prin intermediul subrutinei MAXMAT, apelată de MATOP);

— după ce ordonează crescător elementele vectorului, subrutina ORDVCT nu mai este folosită;

— subrutinele SUMMAT și MAXMAT sînt folosite de subrutina MATOP, dar nu în același timp.

Aceste observații ne conduc la alegerea următoarei structuri de acoperire pentru segmentele de program, dată prin cartela TREE (numele segmentelor apare în afara parantezelor, iar cele ale unităților de program care sînt cuprinse în ele apar în interiorul parantezelor):



Segmentul de date comune F%BLK va fi încărcat în continuarea segmentului de program PRINC; segmentul de date comune BETA va fi încărcat în continuarea segmentului de program A; segmentul de date MAXIM va fi încărcat în continuarea segmentului de program cu același nume.

În urma editării legăturilor, la imprimantă sînt tipărite două liste.

Prima listă cuprinde numele segmentelor, numele modulelor din care sînt compuse și adresele de implantare în memorie (de care nu ne vom ocupa aici). Deoarece segmentul PRINC care conține programul principal are în componență multe rutine ale sistemului, vom da în continuare numai lista celorlalte segmente:

```
SEGMENT   F%BLK     NO   2
          MODULE    F%BLK
          MODULE    ALFA
SEGMENT   PAR       NO   3
          MODULE    TSTPAR
          MODULE    F%VMOD
SEGMENT   MAXIM     NO   4
          MODULE    MAX
SEGMENT   MAXIM     NO   5
          MODULE    DATAMAX
SEGMENT   OVCT      NO   6
          MODULE    ORDVCT
SEGMENT   A         NO   7
          MODULE    MATOP
SEGMENT   BETA      NO   8
          MODULE    BETA
SEGMENT   B         NO   9
          MODULE    SUMMAT
SEGMENT   C         NO   10
          MODULE    MAXMAT
```

A doua listă urmează după mesajul:

IMPLANT. APRES TRAITEMENT OPTION FMS

care arată că au fost incorporate și acele module ale sistemului de gestiune a fișierelor de care este nevoie. În această listă apar numele segmentelor și adresele de implantare în

memorie relative la adresa de la care începe încărcarea în memorie a segmentelor. Lista este următoarea:

```
SEGMENT    PRINC     NO   1    IMPLANTATION     0
                     LONGUEUR DU SEGMENT  3C28
SEGMENT    F%BLK     NO   2    IMPLANTATION  3C28
                     LONGUEUR DU SEGMENT    38
SEGMENT    PAR       NO   3    IMPLANTATION  3C60
                     LONGUEUR DU SEGMENT   100
SEGMENT    MAXIM     NO   4    IMPLANTATION  3C60
                     LONGUEUR DU SEGMENT    88
SEGMENT    MAXIM     NO   5    IMPLANTATION  3CE8
                     LONGUEUR DU SEGMENT    28
SEGMENT    OVCT      NO   6    IMPLANTATION  3D10
                     LONGUEUR DU SEGMENT    E0
SEGMENT    A         NO   7    IMPLANTATION  3D10
                     LONGUEUR DU SEGMENT   220
SEGMENT    BETA      NO   8    IMPLANTATION  3F30
                     LONGUEUR DU SEGMENT    60
SEGMENT    B         NO   9    IMPLANTATION  3F90
                     LONGUEUR DU SEGMENT   150
SEGMENT    C         NO  10    IMPLANTATION  3F90
```

Din adresele de încărcare în memorie se poate verifica cu ușurință că este obținută într-adevăr structura de acoperire precizată în cartela TREE.

Capitolul IX

FACILITĂȚI DE RULARE A PROGRAMELOR ÎN SISTEMUL SIRIS

Vom relua prezentarea fazelor prin care trece un program, făcînd o descriere mai amănunțită a posibilităților existente în sistemul SIRIS.

COMPILAREA

Intrarea pentru compilator o constituie programele sursă. Acestea se pot afla sau în fișierul de lucrări, sau într-o bibliotecă sursă.

Dacă trebuie compilat un program care se află în fișierul de lucrări, imediat înaintea acestui program trebuie plasată cartela de comandă COMPILE.

Dacă programul care trebuie compilat se află într-o bibliotecă sursă, atunci cartela COMPILE trebuie urmată imediat în fișierul de lucrări de un ordin FETCHS care identifică modulul sursă care trebuie compilat. Forma ordinului este următoarea:

* FETCHS LN: *nume*, DV: *perif* $\left[,\left\{\begin{array}{l}\text{CD: } dată\\ \text{GN: } g\end{array}\right\}\right]$

[, VN: *v*], FN: *carte* [, UN: *u*]

Coloana 1 conține un asterisc; coloanele 2 și 9 conțin cîte un blanc; coloanele 3—8 conțin identificatorul FETCHS.

Lista de argumente începe din coloana 10 și poate continua pînă cel mult în coloana 71. Cartela FETCHS nu admite o cartelă de continuare. În lista de argumente nu pot apare blancuri.

nume este numele bibliotecii sursă; *carte* este numele cărții care urmează să fie compilată; celelalte argumente au fost prezentate la studiul fișierelor.

În urma compilării unei unități de program se obține un modul obiect. Sistemul permite utilizatorului ca după compilare modulul obiect să fie:

1) — înregistrat în fișierul de intrare al editorului de legături, ceea ce permite ca să fie editat împreună cu alte module obiect;

2) — perforat pe cartele în vederea unei folosiri ulterioare;

3) — catalogat într-o bibliotecă utilizator B.T. de unde poate fi preluat ori de câte ori este nevoie.

Rezultatul compilării este totdeauna înscris în fișierul FILEDIT.

Ieșirea compilatorului este precizată prin informațiile date de utilizator în cartela COMPILE. Forma generală a acestei cartele este următoarea:

COMPILE *ident* [, *listă de opțiuni*] [,LN: *lib*, FN: *nume*, DV: *perif* [,VN: *v*] [, {CD: *dată* / GN: *g*}]] [,PUN]

Semnificația argumentelor a fost prezentată fie în forma restrînsă a cartelei COMPILE, fie în ordinul FETCHS.

Prin folosirea opțiunii PUN se realizează și perforarea modulului obiect pe cartele. Pachetul de cartele obținut poate fi folosit ulterior într-o altă lucrare pentru a fi editat împreună cu alte module obiect; în acest scop se folosește cartela BINARY, care trebuie să fie plasată imediat înaintea pachetului de cartele. Cartela BINARY informează monitorul că informațiile care urmează constituie un modul obiect care trebuie preluat ca atare de editorul de legături în procesul de editare.

O altă utilizare a unui modul obiect perforat pe cartele o constituie introducerea sa într-o bibliotecă utilizator cu ajutorul cartelei INCLUDE.

Pentru catalogarea modulului obiect rezultat în urma compilării într-o bibliotecă B.T., fără a obține în prealabil acest modul obiect pe cartele, trebuie folosite argumentele care apar în cartela COMPILE după *listă de opțiuni*.

EDITAREA LEGĂTURILOR

Intrarea pentru editorul de legături este constituită de modulele obiect aflate în fișierul FILEDIT. Un modul obiect poate fi adus în acest fișier fie prin intermediul unei cartele COMPILE, fie printr-una dintre cartelele de comandă BINARY sau FETCHB.

Cartela BINARY permite aducerea în fișierul FILEDIT a unui modul obiect care se află în fișierul de lucrări imediat după această cartelă. Forma ei este următoarea:

• BINARY [*identif*]

unde *identif* este numele pe care avem posibilitatea să-l atribuim modulului obiect; *identif* este format din 1—8 caractere alfanumerice, dintre care primul este alfabetic.

Cartela FETCHB permite copierea în fișierul FILEDIT a unui program în format B.T. catalogat într-o bibliotecă utilizator. Forma ei este următoarea:

• FETCHB LN: *nume*, FN: *carte*, DV: *perif*,

$$\left\{\begin{array}{l}\text{CD: } data\\ \text{GN: } g\end{array}\right\}\text{,VN: } v\text{ [,UN: } u\text{] [,FS: } f\text{]}$$

unde toate argumentele au fost prezentate anterior.

Dacă argumentul UN lipsește, este ales acel modul obiect cu numele *carte* care are cel mai mare număr de actualizare.

Atît cartela BINARY, cît și cartela FETCHB trebuie să apară înaintea cartelei LINK.

În general, ieșirea pentru editorul de legături o constituie fișierul de ieșire al acestuia, fișier care are numele LOADGO. Alte două posibilități sînt catalogarea programului în format I.M.T. într-o bibliotecă sau perforarea lui pe cartele. Alternativa aleasă este indicată de utilizator cu ajutorul argumentelor din cartela LINK. Forma generală a acestei cartele este următoarea:

• LINK [LN:*nume*, $\left\{\begin{matrix}\text{CD}:\textit{dată}\\ \text{GN}:g\end{matrix}\right\}$, VN:*v*, DV: *perif*, FN: *carte*] [,PUN] [,ER:*n*]

Prezența argumentului PUN determină perforarea programului în format I.M.T. pe cartele.

Pentru catalogarea programului obiect într-o bibliotecă I.M.T. vor fi folosite argumentele FN, LN, DV, VN, precum și unul dintre argumentele CD, GN. Menționăm că suportul bibliotecii nu poate fi decît un disc, de altfel ca și în cazul catalogării unui modul obiect într-o bibliotecă cu ajutorul cartelei COMPILE.

Deci pentru catalogarea unui program I.M.T. într-o bibliotecă utilizator putem proceda într-unul dintre următoarele moduri:

1) — se obține programul I.M.T. pe cartele și apoi se introduce în bibliotecă folosind cartela de comandă INCLUDE a bibliotecarului;

2) — se introduce în bibliotecă un program obiect care se află deja catalogat în altă bibliotecă (prin includere, înlocuire sau prin copierea întregii biblioteci);

3) — este catalogat direct programul I.M.T. obținut în urma editării legăturilor, specificînd acest lucru prin cartela LINK.

LANSAREA UNUI PROGRAM

Pentru lansarea unui program se folosește cartela RUN. Programul obiect trebuie să se afle sau în fișierul LOADGO, sau în biblioteca B.U.S. sau într-o bibliotecă I.M.T. utilizator.

Dacă programul se află în fișierul LOADGO, vom folosi o cartelă RUN în care nu apare argumentul FN; prin aceasta programul este încărcat în memorie și este lansată execuția

lui. Menționăm că după încărcarea în memorie a programului obiect din fișierul LOADGO, programul nu este șters din acest fișier, ceea ce permite executarea lui de mai multe ori în cadrul aceleiași lucrări. Începînd însă cu a doua execuție, programul trebuie adus explicit în memorie în prealabil (printr-o cartelă FETCH) și apoi putem lansa noua execuție.

Dacă programul obiect se află în biblioteca B.U.S., pentru încărcarea sa în memorie și lansarea execuției sale trebuie folosită o cartelă RUN în care apare argumentul FN conținînd numele sub care este catalogat programul în bibliotecă.

Dacă programul obiect se află într-o bibliotecă I.M.T. utilizator, sînt necesare o cartelă FETCH pentru încărcarea sa în memorie și apoi o cartelă RUN (în care este prezent argumentul FN) pentru lansarea lui.

Cartela FETCH are următoarea formă:

• FETCH LN:*nume*[,DV:*perif*] [,{CD:*dată* / GN:*g*}]

[,VN:*v*] [,FN:*carte*] [,UN:*u*]

unde *nume* poate fi:

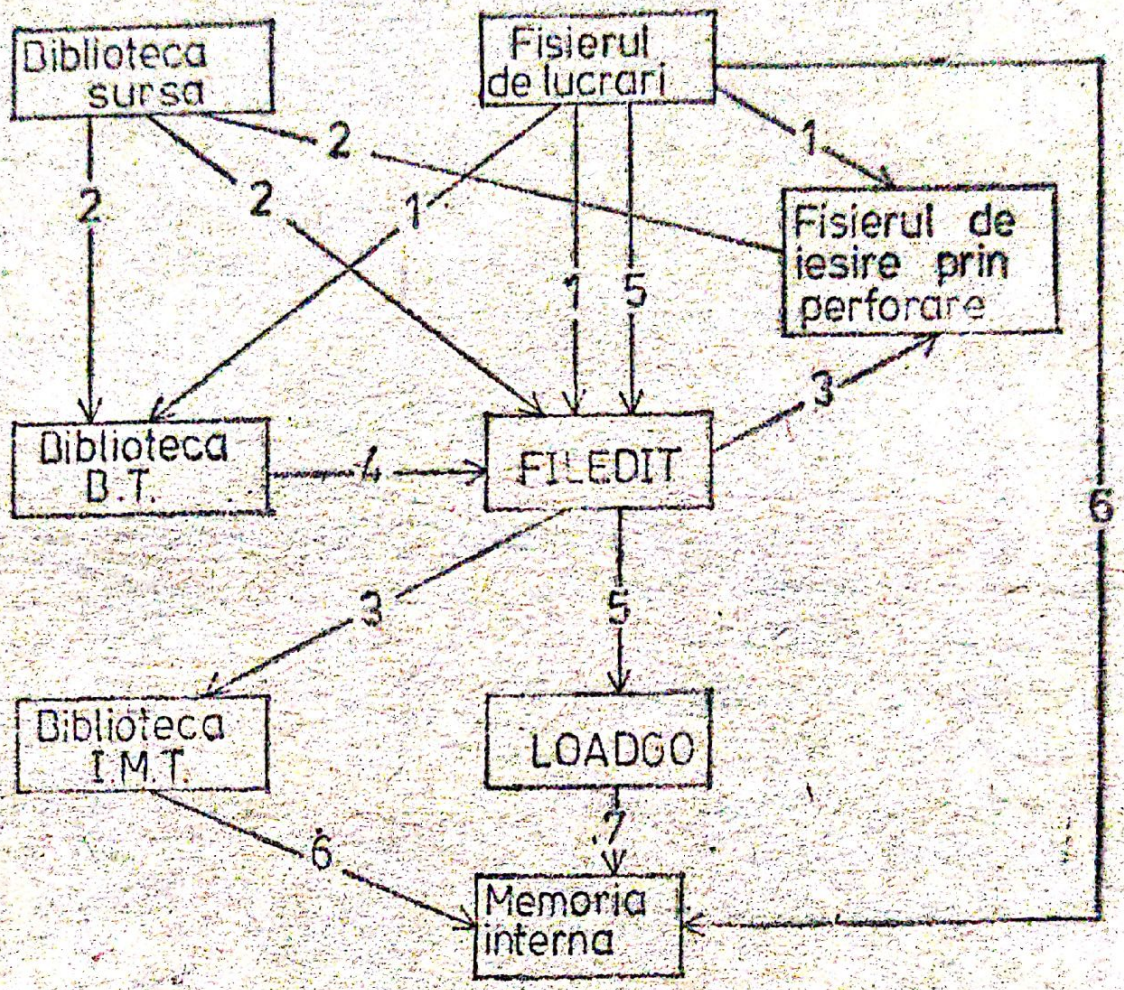
1) — *1 (ceea ce indică faptul că programul obiect se află în fișierul de lucrări imediat după cartela FETCH)

2) — *4 (caz în care programul se află în fișierul LOADGO)

3) — numele unei biblioteci care conține programul a cărui execuție trebuie lansată; programul este identificat prin intermediul celorlalte argumente.

Facem observația că dacă programul I.M.T. se află într-o bibliotecă utilizator, aducerea lui în memoria internă este posibilă numai dacă suportul bibliotecii este un disc.

Un rezumat al posibilităților de rulare a unui program apare în următoarea schemă:



unde:

1 COMPILE
2 FETCHS + COMPILE
3 LINK
4 FETCHB
5 BINARY
6 FETCH
7 FETCH sau RUN

Exemplu

Vom aplica unele dintre facilitățile de rulare descrise pentru rularea programului din capitolul precedent.

Prima lucrare este următoarea:

```
.       JOB CREARE,AN:CCUB,PN:HOPE
.       INIT DV:RD3,VS:111111
.       ALLOC DV:RD3,VN:1,GN:1,FN:'BIBSOUP SOU',          *
.             AM:ANY,SZ:10
.       ALLOC DV:RD3,FN:'BIBBTP  RBN',GN:1,VN:1,          *
.             AM:ANY,SZ:10
.       ALLOC DV:RD3,FN:'BIBIMT  IMT',GN:1,VN:1,          *
.             AM:ANY,SZ:10
.       SYSRUN BIBLIO
% OPNLIB S,DV:RD3,LN:BIBSOUP,GN:1,VN:1,FT:SOU,INIT
% OPNLIB R,DV:RD3,LN:BIBBTP,GN:1,VN:1,FT:RBN,INIT
% OPNLIB C,DV:RD3,LN:BIBIMTP,GN:1,VN:1,FT:IMT,INIT
% INCLUDE OL:*1,OF:SMAIN,DL:S
      programul principal
% INCLUDE OL:*1,OF:SPAR,DL:S
      subrutina TSTPAR
% INCLUDE OL:*1,OF:SMAXIM,DL:S
      subrutina MAX
% INCLUDE OL:*1,OF:SOVCT,DL:S
      subrutina ORDVCT
% INCLUDE OL:*1,DL:S,OF:SA
      subrutina MATOP
% INCLUDE OL:*1,DL:S,OF:SB
      subrutina SUMMAT
% INCLUDE OL:*1,DL:S,OF:SC
      subrutina MAXMAT
% ENDLIB
.       EOJ
```

În această lucrare au fost create o bibliotecă sursă cu numele BIBSOUP, o bibliotecă B.T. cu numele BIBBTP și o bibliotecă I.M.T. cu numele BIBIMTP. Apoi toate unitățile de program au fost catalogate în biblioteca sursă creată.

Scanned with OKEN Scanner

A doua lucrare este următoarea:

```
.       JOB CAT,AN:CCUB,PN:HOPE
.       COMPILE FORTRAN,LN:BIBBTP,DV:RD3,VN:1,                    *
.               GN:1,FN:RPAR
* FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SPAR
.       COMPILE FORTRAN,LN:BIBBTP,DV:RD3,VN:1,                    *
.               GN:1,FN:RA
* FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SA
.       COMPILE FORTRAN,LN:BIBBTP,DV:RD3,VN:1,                    *
.               GN:1,FN:RMAXIM
* FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SMAXIM
.       COMPILE FORTRAN,LN:BIBBTP,DV:RD3,VN:1,                    *
.               GN:1,FN:RB
* FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SB
.       COMPILE FORTRAN,LN:BIBBTP,DV:RD3,VN:1,                    *
.               GN:1,FN:RC
* FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SC
.       EOJ
```

Această a doua lucrare realizează catalogarea unor module obiect obținute prin compilarea unor module sursă conținute în biblioteca sursă creată. În urma acestor două lucrări, unitățile de program inițiale se afle în cele două biblioteci sub formă de cărți sursă sau în format B.T. după cum urmează:

Unitate de program	*Bibliotecă*	*Nume*
program principal	BIBSOUP	SMAIN
subrutina TSTPAR	BIBSOUP	SPAR
	BIBBTP	RPAR
subrutina ORDVCT	BIBSOUP	SOVCT
subrutina MAX	BIBSOUP	SMAXIM
	BIBBTP	RMAXIM
subrutina MATOP	BIBSOUP	SA
	BIBBTP	RA

Scanned with OKEN Scanner

subrutina SUMMAT	BIBSOUP	SB
	BIBBTP	RB
subrutina MAXMAT	BIBSOUP	SC
	BIBBTP	RC

A treia lucrare este următoarea:

```
.       JOB EXEC,AN:CCUB,PN:HOPE
.       SEG PRINC
.       COMPILE FORTRAN
. FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SMAIN
.       SEG MAXIM
.       FETCHB LN:BIBBTP,DV:RD3,VN:1,GN:1,FN:RMAXIM
.       SEG PAR
.       FETCHB LN:BIBBTP,DV:RD3,VN:1,GN:1,FN:RPAR
.       SEG OVCT
.       COMPILE FORTRAN
. FETCHS LN:BIBSOUP,DV:RD3,VN:1,GN:1,FN:SOVCT
.       SEG A
.       FETCHB LN:BIBBTP,DV:RD3,VN:1,GN:1,FN:RA
.       SEG B
.       FETCHB LN:BIBBTP,DV:RD3,VN:1,GN:1,FN:RB
.       SEG C
.       FETCHB LN:BIBBTP,DV:RD3,VN:1,GN:1,FN:RC
.       TREE PRINC+(F%BLK+ALFA)(PAR,MAXIM(OVCT,A+BETA(B,C)))
.       LINK LN:BIBIMTP,FN:PROB,DV:RD3,GN:1,VN:1
.       RUN
    date de intrare
.EOF
.       EOJ
```

Această a treia lucrare cuprinde fazele de editare a legăturilor și execuției.

Editarea legăturilor se execută asupra modulelor obiect catalogate în biblioteca BIBBTP (și aduse în fișierul FILEDIT cu ajutorul cartelelor FETCHB) și asupra modulelor

obiect obținute prin compilarea cărților SMAIN și SOVCT din biblioteca sursă BIBSOUP.

Programul obiect obținut este catalogat în biblioteca BIBIMTP.

Execuția are loc pentru aceleași date de intrare ca și în programul precedent, deci și rezultatele vor coincide.

PROCEDURI CATALOGATE

Orice modul (program sursă, BT, IMT) poate fi catalogat într-o bibliotecă de tip corespunzător, putînd fi apelat printr-o singură instrucțiune. De asemenea orice carte conținută într-o bibliotecă poate fi modificată conform necesităților utilizatorului.

Folosind aceste facilități munca programatorului este mult mai ușoară. Prin intermediul instrucțiunilor limbajului de comandă se poate alcătui lucrarea propriu-zisă în care este indicată ordinea și structura prelucrărilor astfel ca în final să se obțină rezultatul dorit. Sistemul de operare SIRIS permite ca grupul cartelelor de comandă împreună cu anumite module necesare să fie catalogat într-o bibliotecă și apelat printr-o comandă. Această facilitate este foarte utilă cînd respectivele cartele de comandă sînt utilizate în mai multe lucrări sau sînt necesare pentru folosire multiplă. În acest fel se realizează o mai mare ușurință în alcătuirea lucrărilor.

Procedurile pot fi catalogate în biblioteca de proceduri catalogate Z%PROC sau în orice bibliotecă sursă a utilizatorului. Introducerea unei cărți într-o bibliotecă de proceduri catalogate, se face la fel ca pentru orice bibliotecă sursă.

O carte din biblioteca de proceduri catalogate conține în general cartele de comandă (altele decît •JOB și •EOJ) și cînd este cazul chiar programe (de exemplu o cartelă •COMPILE urmată de programul sursă care trebuie compilat etc.)

Procedura catalogată va fi folosită în momentul cînd în fișierul de lucrări este întîlnită cartela de comandă •XPROC; în această situație efectul este același ca în cazul în care carte-

tele de comandă din cartea citată de cartela XPROC s-ar afla în fișierul de lucrări. În plus este posibil de a modifica pentru moment cartelele din procedura catalogată, obținînd astfel o și mai mare flexibilitate. De asemenea este posibil ca o procedură catalogată să conțină la rîndul ei o instrucțiune XPROC, obținîndu-se astfel o structurare a lucrărilor (facem însă observația că este exclusă recursivitatea).

Mai exact, lucrurile decurg astfel: la întîlnirea unei cartele de comandă XPROC (prin care se cere executarea unei proceduri catalogate), monitorul de înlănțuiri întrerupe exploatarea fișierului de lucrări (în care se află și cartela XPROC) executînd în continuare procedura identificată prin această cartelă. Procedura este trecută din Z%PROC (sau din biblioteca utilizator) corespunzătoare într-un fișier asemănător cu fișierul de lucrări numit Z%SYSFT; în continuare acest fișier este substituit fișierului de lucrări pe perioada executării procedurii, după care se revine la fișierul de lucrări inițial.

În continuare trecem la descrierea cartelelor de comandă necesare lucrului cu proceduri catalogate.

1) — *Cartela XPROC* are următoarea formă:

- XPROC { *carte*[,UN:*u*] / *,DV:*perif*,FN:'*nume/carte*'[,GN:*g*][,VN:*v*] } [,MOD] [,DEF]

Prima alternativă se folosește cînd procedura este catalogată în Z%PROC; *carte* reprezintă numele fișierului (cărții), iar *u* numărul de actualizare.

A doua alternativă se folosește cînd procedura este catalogată într-o bibliotecă utilizator; *nume* este numele bibliotecii, *carte* este numele cărții, *g* este numărul de generație, iar *v* este numărul de versiune.

O procedură catalogată poate fi modificată pe timpul execuției ei dacă se folosește opțiunea MOD (în caz contrar este executată procedura în forma în care apare în bibliotecă). Folosirea opțiunii MOD este legată de folosirea cartelelor de comandă MOD și ENDMOD, care vor fi prezentate ulterior.

În anumite cazuri — de exemplu punerea la punct a procedurilor — este posibilă introducerea lor în lucrarea respectivă, urmînd ca orice apelare ulterioară a acesteia să se facă conform regulilor stabilite. În acest caz este folosită opțiunea DEF, iar procedura urmează imediat cartelei XPROC. În această situație, procedura trebuie să fie urmată imediat de cartela de comandă ENDPROC.

2) — *Cartela de comandă MOD* permite efectuarea de modificări temporare într-o procedură catalogată într-o bibliotecă. Pentru aceasta cartela XPROC este urmată de seturi de cartele; fiecare set de cartele începe cu cartela MOD și continuă cu cartelele care trebuie inserate în procedură sau care înlocuiesc o secvență de cartele din procedură; în mod normal sfîrșitul acestor cartele este delimitat de următoarea cartelă MOD sau de cartela ENDMOD. Forma cartelei MOD este următoarea:

MOD n_1 [,n_2] [,CN: n_3]

Dacă apare doar n_1, cartelele care urmează lui MOD vor fi adăugate în procedura catalogată după cartela cu numărul de secvență n_1.

Dacă apar atît n_1 cît și n_2, cartelele cu numerele de secvență n_1, n_1+1...,n_2 din procedura catalogată vor fi înlocuite cu cartelele care urmează lui MOD.

Pentru aceste două situații, sfîrșitul cartelelor modificatoare este cel normal, adică cel descris mai sus.

Folosirea opțiunii CN permite să nu se facă verificarea dacă în cele n_3 cartele care urmează după cartela MOD se află cartele care sînt de tipul MOD sau ENDMOD, fiind creată astfel posibilitatea inserării unei noi secvențe de modificare în textul procedurii catalogate. Prin aceasta în urma introducerii cratelelor de modificare în locul unde era apelată o altă procedură catalogată, aceasta va fi modificată conform secvenței ce a fost inserată.

3) — *Cartela ENDMOD* indică sfîrșitul modificărilor temporare într-o procedură catalogată. Forma ei este:

ENDMOD

4) — *Cartela ENDPROC* delimitează sfîrșitul unei proceduri, în situația în care în cartela XPROC apare opțiunea DEF. Forma ei este următoarea:

```
.          ENDPROC
```

Exemplu

Să presupunem că au fost executate primele două lucrări din exemplul precedent din acest capitol. Considerăm acum următoarea a treia lucrare:

```
.          JOB HP,AN:CCUB,PN:HOPE
.          ALLOC DV:RD3,FN:'BIBPROC SOU',GN:1,VN:1,
.                AM:ANY,SZ:10
.          SYSRUN BIBLIO
% OPNLIB P,DV:RD3,LN:BIBPROC,GN:1,VN:1,FT:SOU,INIT
% INCLUDE OL:*1,OF:PROC1,DL:E
      secvența de cartele din lucrarea a treia din exemplul
      precedent,secvență care începe cu prima cartelă SEG
      și se încheie cu cartela LINK
% ENDLIB
.          EOJ
```

Prin această lucrare am creat o bliblioteca de proceduri catalogate, în care am introdus o carte cu numele PROC1 conținînd secvența de cartele de comandă amintite.

Pentru rularea programului este acum suficient să considerăm lucrarea următoare:

```
.          JOB RUL,AN:CCUB,PN:HOPE
.          XPROC *,DV:RD3,FN:'BIBPROC SOU/PROC1',VN:1,GN:1
.          RUN
    date de intrare
,EOF
.          EOJ
```

Să presupunem acum că la o rulare ulterioară a programului dorim să obținem și perforarea pe cartele a modulului obiect

rezultate prin compilarea cărții cu numele SMAIN din biblioteca BIBSOUP. Pentru aceasta este suficient să scriem lucrarea:

```
.JOB MOD,AN:CCUB,PN:HOPE
.XPROC *,DV:RD3,FN:'BIBPROC SOU/PROC1',
        VN:1,GN:1,MOD
.MOD 2,2
.COMPILE FORTRAN,PUN
.ENDMOD
.RUN
date de intrare
.SOF
.EOJ
```

Observație Cînd procedura catalogată face parte din lucrare nu trebuie perforat nimic în coloanele 73—80.

LUCRUL CU BIBLIOTECI DE SUBPROGRAME TRANSLATABILE

Sistemul de operare SIRIS folosește două tipuri de biblioteci care conțin module obiect și anume:

— biblioteci B.T.

— biblioteci de subprograme translatabile (biblioteci SP).

La editarea legăturilor, pentru fiecare carte folosită dintr-o bibliotecă B.T., utilizatorul trebuie să scrie o cartelă de comandă FETCHB.

Subprogramele translatabile din biblioteca SP sînt aduse automat în acest fișier dacă în celelalte module obiect s-a făcut o referire la numele unității de program care le-a generat prin compilare (deci nu mai trebuie folosită cartela FETCHB). Astfel utilizatorul este scutit de sarcina de a indica explicit, prin cartele de tipul FETCHB, subprogramele folosite.

Această sarcină este preluată de editorul de legături.

Bineînțeles că în momentul editării legăturilor, subprogramele translatabile necesare trebuie să fi fost anterior catalogate în biblioteca SP. Singura obligație a utilizatorului este să valideze accesul editorului de legături la biblioteca respectivă.

Lucrul cu biblioteci de subprograme translatabile constă deci în următoarele:

1) — *Catalogarea într-o bibliotecă SP*

După cum am văzut, introducerea unei cărți într-o bibliotecă B.T. se poate face în următoarele moduri:
— în urma compilării, prin cartela COMPILE;
— de către programul bibliotecar prin copiere, includere sau înlocuire (folosind deci cartelele de comandă ale bibliotecarului COPY, INCLUDE, REPLACE);

În cazul bibliotecilor SP, singurele posibilități de includere a unei cărți sînt următoarele:
— prin copierea dintr-o bibliotecă de același tip;
— prin folosirea cartelei de comandă a bibliotecarului RELIB.

Cartela RELIB permite introducerea, ștergerea sau înlocuirea unui subprogram translatabil într-o bibliotecă SP. Forma ei este următoarea:

$$\%\ \text{RELIB}\ \left\{\begin{matrix}\text{INCL}\\ \text{DELT}\\ \text{REPL}\end{matrix}\right\},carte,\text{DL}:a_1[,\text{OL}:a_2][,\text{ON}:u]$$

Argumentele INCL, DELT, REPL determină corespunzător includerea, ștergerea sau înlocuirea unei cărți.

carte este numele cărții.

a_1 este idexul bibliotecii asupra căreia se face prelucrarea.

a_2 este idexul bibliotecii de origină sau *1 în cazul fișierului de lucrări.

u este numele de actualizare al cărții prelucrate.

2) — *Modul de folosire al bibliotecilor SP*

Pentru a valida accesul editorului de legături la bibliotecă, este necesară folosirea cartelei LIB. Forma ei este următoarea:

```
LIB LN:nume,VN:v,GN:g,DV:perif
```

unde argumentele identifică biblioteca SP. Forma argumentelor este cea cunoscută. Dacă este utilizată această cartelă, pentru fiecare instrucțiune din program de tipul CALL (chemarea unui subprogram), editorul de legături îl va căuta în bibliioteca indicată și-l va prelua.

Exemplu

Să considerăm un program FORTRAN format din următoarele unități de program;

— programul principal (care apelează subrutina S1);
— subrutina S1 (care apelează subrutina S2);
— subrutina S2.

Exemplul de față este format din 4 lucrări, după cum urmează:

1) *Crearea bibliotecilor*

```
        JOB INIT,AN:CCUB,PN:HOPE
        INIT DV:RD3,VS:111111
        ALLOC DV:RD3,FN:'BIBBT   RBN',VN:1,GN:1,AM:ANY,SZ:10
        ALLOC DV:RD3,FN:'BIBSP   RSL',VN:1,GN:1,AM:ANY,SZ:10
        SYSRUN BIBLIO
% OPNLIB B,DV:RD3,LN:BIBBT,GN:1,VN:1,FT:RBN,INIT
% OPNLIB P,DV:RD3,LN:BIBSP,GN:1,VN:1,FT:RSL,INIT
% ENDLIB
        EOJ
```

2) *Catalogarea în biblioteca B.T.*

```
        JOB CATBT,AN:CCUB,PN:HOPE
        COMPILE FORTRAN,LN:BIBBT,DV:RD3,VN:1,GN:1,FN:R1
    subrutina S1
        COMPILE FORTRAN,LN:BIBSP,DV:RD3,VN:1,GN:1,FN:R2
    subrutina S2
        EOJ
```

Numele cărților din biblioteca B.T. corespunzătoare unităților de program S1 și S2 sînt R1 și R2.

3) *Catalogarea în bibliotcca SP*

```
.       JOB CATSP,AN:CCUB,PN:HOPE
.       SYSRUN BIBLIO
% OPNLIB B,DV:RD3,GN:1,VN:1,FT:RBN,LN:BIBBT
% OPNLIB P,DV:RD3,GN:1,VN:1,FT:RSL,LN:BIBSP
% RELIB INCL,R1,DL:P,OL:B
% RELIB INCL,R2,DL:P,OL:B
% ENDLIB
.       EOJ
```

În urma acestei lucrări, biblioteca SP va conține subprogramele translatabile R1 și R2, corespunzătoare subrutinelor S1 și S2.

4) *Utilizarea subprogramelor translatabile catalogate*

```
.       JOB EXEC,AN:CCUB,PN:HOPE
.       COMPILE FORTRAN
      programul principal
.       LIB LN:BIBSP,VN:1,GN:1,DV:RD3
.       LINK
.       RUN
      date de intrare
.EOF
.       EOJ
```

Ca urmare a prezenței cartelei LIB, în momentul editării legăturilor modulele obiect corespunzătoare unităților de program S1 și S2 vor fi aduse în FILEDIT datorită referirilor care sînt făcute la numele lor.

Dacă ar fi lipsit cartela LIB, la editarea legăturilor ar fi apărut un mesaj indicînd imposibilitatea referirilor la numele S1 și S2.

Scanned with OKEN Scanner

Capitolul X

DEPANAREA PROGRAMELOR

Printre serviciile asigurate utilizatorului de sistemul de operare SIRIS 3 se numără și aceea de asistență în depanarea programelor. Astfel sistemul nu se limitează la a-l anunța pe utilizator că lucrarea sa nu poate fi executată, dar îi și indică cît mai precis cu putință cauza care stă la baza imposibilității execuției.

Astfel, dacă utilizatorul va folosi în programul său funcția standard ALOG10 dînd ca argument un număr negativ, sistemul de operare va anunța că nu poate executa operația deoarece argumentul este negativ, deci nu face parte din domeniul de definiție al funcției logaritm; utilizatorul are posibilitatea să detecteze, folosind informațiile furnizate de sistemul de operare, instrucțiunea din programul său care conține apelarea eronată a funcției logaritm.

Pentru utilizatorul care folosește limbajul FORTRAN, sistemul de operare depistează erori în toate cele trei faze de rulare a programelor: compilare, editare de legături, execuție.

ERORI DEPISTATE LA COMPILARE

Orice încălcare a regulilor de sintaxă ale limbajului FORTRAN este depistată de către compilator, care produce un mesaj de forma:

** LIGNE *nr:instrucțiune*
NIVEAU *n:mesaj*

unde:

nr — numărul liniei din unitatea de program care conține instrucțiunea eronată

instrucțiune — instrucțiunea respectivă

n — gravitatea erorii

mesaj — text explicativ descriind tipul erorii comise

Gravitatea erorii este indicată printr-un nivel de eroare cuprins între 1 și 4.

nivelul 1 — erori ale căror cauze sînt evidente; compilatorul trece la corectarea lor. De exemplu dacă într-o expresie aritmetică lipsește o paranteză închisă, compilatorul presupune că ea trebuie să se afle la sfîrșitul expresiei;

nivelul 2 — erori mai importante, dar pentru care compilatorul încearcă totuși să genereze instrucțiunile în cod mașină corespunzătoare. De exemplu dacă o variabilă a fost declarată de două ori, a doua declarare nu va mai fi luată în considerare;

nivelul 3 — erori grave. Compilatorul nu produce nimic pentru instrucțiunile care le conțin. De exemplu folosirea în instrucțiunea DO a unei variabile reale ca variabilă DO reprezintă o eroare de nivelul 3.

nivelul 4 — erori foarte grave. În asemenea situații compilarea programului este abandonată. O eroare de nivel 4 apare de exemplu dacă un bloc de comun depășește 64 Kocteți.

În mod normal, editarea legăturilor nu se poate face dacă la compilare a fost detectată o eroare de nivel mai mare sau egal cu 2. Totuși utilizatorul are posibilitatea să aleagă nivelul *n* de eroare prin opțiunea ER:*n* din cartela de comandă COMPILE FORTRAN. Dacă în program apar la compilare erori de nivel mai mic decît *n*, se va putea încerca editarea legăturilor.

Observații:

1) O eroare de nivel 4 va interzice editarea legăturilor. Rezultatul compilării nu va putea fi catalogat într-o bibliotecă B.T. dacă modulul sursă conține erori de sintaxă. Mesajele care apar în asemenea situații sînt următoarele:

COMPILATION ABANDONNEE
MISE EN CATALOGUE NON EFECTUEE

2) — La sfîrșitul acțiunii compilatorului este tipărit un mesaj de forma:

*** FIN DE COMPILATION (PLUS HAUT NIVEAU D'ERREUR RENCONTREE = *n*) hh. mm. ss.

unde ultima informație reprezintă timpul curent.

3) — Deoarece mesajele ce indică erori descoperite în timpul compilării sînt suficient de explicite, utilizatorul reușind pe baza lor să descopere eroarea comisă, nu vom da aici lista lor.

ERORI SEMNALATE LA EDITAREA LEGĂTURILOR

Editarea legăturilor este de la început abandonată în cazul cînd nivelul maxim al erorilor detectate în cursul compilării este mai mare sau egal cu cel din opțiunea ER din cartela COMPILE FORTRAN (în cazul nespecificării lui ER, nivelul este 2). În acest caz este tipărit mesajul:

EDITION DE LIENS IMPOSSIBLE* ABANDON DU TRAVAIL

În caz contrar se încearcă să se efectueze editarea legăturilor. Orice eroare detectată în faza de editare a legăturilor este semnalată, printr-un mesaj de forma:

** ERREUR *nnnn mesaj*

unde *nnnn* conține pe prima poziție nivelul de gravitate al erorii, iar pe celelalte poziții numărul erorii; *mesaj* descrie natura erorii.

Scanned with OKEN Scanner

Un exemplu de eroare de nivel 2 este apariția a două ordine SEG ce definesc același segment; al doilea este ignorat. Constituirea unui segment care necesită o memorie mai mare decît 64 Kocteți reprezintă o eroare de nivel 4.

O eroare de nivel 4 va întrerupe în orice caz editarea legăturilor.

ERORI CARE ÎMPIEDICĂ EXECUȚIA

Este vorba de erori efectuate în fazele anterioare execuției și care fac ca tentativa de lansare a programului să eșueze. Corespunzător fiecărui caz, va fi tipărit unul dintre mesajele:

1) EXECUTION IMPOSSIBLE* ABANDON DU TRAVAIL

Acest mesaj apare în situația în care nivelul de eroare în editarea legăturilor (specificat prin opțiunea ER: *n* din cartela LINK; prin lipsă *n*=2) este mai mare sau egal cu *n*.

2) IMPOSSM — zona de memorie disponibilă este insuficientă pentru încărcarea programului.
3) IMPOSSL — în timpul editării legăturilor nu au fost încorporate toate modulele S.G.F. necesare
4) IMPOSSD — nu a fost făcută asignarea unor periferice
5) IMPOSSN — programul nu a fost găsit în biblioteca B.U.S.
6) IMPOSSF — fișierul FILECOM este saturat
7) IMPOSSC — una dintre cartelele de comandă INIT, FILE, ALLOC, LABEL este incorectă.

ERORI LA EXECUȚIE

Acestea sînt erorile care apar în timpul execuției programului și care împiedică continuarea lui. Ele se pot împărți în 6 categorii. Fiecărei categorii îi este propriu un mesaj care arată natura erorii și (numai în unele cazuri) permite identificarea instrucțiunii din program a cărei executare a condus la eroarea respectivă.

1) — Mesajul:

DEMANDE INSATISFAITE XXXXXXXX

apare pentru orice apel supervizor nesatisfăcut.

Exemplu.

Într-un program utilizatorul dorește să afișeze la consolă un mesaj care pe lîngă un anumit mesaj să indice și adresa fizică a dispozitivului pe care a fost montat un anumit volum. (Acest lucru poate fi realizat folosind TYPE cu un bloc de comandă de tipul TWB, C.) În acest caz adresa fizică a periphericului este găsită de către sistemul de operare pe baza unei informații conținute în zona cu textul mesajului, și care poate fi completată de utilizator. Dacă această informație este eronată lansarea operației de scriere nu va conduce la găsirea unui dispozitiv fizic corespunzător acelei informații eronate, deci se va declanșa afișarea mesajului DEMANDE INSATISFAITE.

2) *Terminarea execuției la cererea operatorului*

În momentul în care operatorul intervine pentru a întrerupe executarea programului, la fișierul de ieșire apar mesajele:

DECISION OPERATEUR XXXXXXXX
INTERVENTION OPERATEUR

după care poate apare vidajul memoriei conform opțiunilor din cartela RUN.

3) — *Erori de intrare/ieșire ireparabile*

Sistemul detectează erorile de acest tip, tipărește mesajul:

ERREUR E/S YY XXXXXXXX

și determină vidajul memoriei. Execuția programului este terminată.

În cadrul restrictiv al acestui manual, informații suplimentare pot fi date numai de codul YY, ale cărui valori descriu natura erorii. Cel mai frecvent apar următoarele coduri:

03 — periferic neafectat
04 — adresa zonei tampon pentru fișierul *1 este eronată

Scanned with OKEN Scanner

05 — eroare de programare pe disc
06 — depășirea sfîrșitului de bandă
07 — numărul logic al perifericuluí este nepermis
10 — a fost citită cartela JOB de către utilizator
11 — tentativă de acces pe un disc înainte de deschiderea fișierului
12 — depășirea numărului de linii la imprimantă sau de cartele la perforatorul de cartele (număr stabilit prin cartela RUN)
13 — depășirea zonei de pe disc
15 — violarea protecției scrierii pe bandă
17 — incident hardware pe bandă
20 — incident hardware pe disc
25 — sector inexistent pe disc DIMAS

4) — *Incidente detectate de S.G.F.*

Este vorba de erori care apar la operațiile de citire/scriere. În aceste situații este tipărit mesajul:

id EGF *oo cc mesaj*

unde

id — idexul fișierului
cc — codul erorii
mesaj — mesaj explicativ
oo — cod care indică originea erorii. Aceasta poate fi detectată de modulele de acces S.G.F. sau de modulul OPEN/CLOSE, după cum $oo \in [40, \ldots, 89]$ sau nu. Corespunzător se va căuta codul de eroare *cc* în tabelul erorilor detectate de modulele de acces S.G.F. sau în tabelul erorilor detectate de funcțiile S.G.F. — monitor (vezi anexa 3)

5) — *Incidente detectate la apelarea funcțiilor standard*

Este vorba de erori care apar la apelarea incorectă a acestor funcții (de exemplu, extragerea unui radical dintr-un număr negativ). Aceste erori provoacă tipărirea unui mesaj și terminarea programului. Mesajul are forma:

ERREUR: *nume* (ADRESSE: XXXXXX) *mesaj*;

nume este numele subprogramului apelat;
mesaj arată cauza producerii erorii;

XXXXXX este adresa din memorie a instrucțiunii în cod mașină care a produs eroarea. Cu ajutorul acestei adrese putem stabili instrucțiunea din programul FORTRAN care conduce la eroarea respectivă. Modul în care se poate realiza acest lucru va fi descris ulterior.

6) — *Apariția unei devieri*

În această situație este listat un mesaj, se face vidajul memoriei și se întrerupe execuția programului.
Mesajul are forma:

ERREUR PROGRAMME OOOXXXX

unde valorile luate de XXXX corespund următoarelor situații:

0101 — depășire binară;
0102 — depășire zecimală;
0104 — depășire inferioară în virgulă mobilă;
0108 — depășire superioară în virgulă mobilă (inclusiv împărțire cu 0);
0201 — instrucțiune privilegiată;
0202 — adresă inexistentă;
0204 — violarea unei zone de memorie protejate;
0208 — deviere provocată de ceasul sistemului;
0400 — instrucțiune inexistentă;
0800 — excepție software (eroare de sistem).

Adresa instrucțiunii în cod mașină care a produs eroarea se poate afla în acest caz din cuvîntul de stare al programului (unde apare în al treilea și al patrulea octet).

DESCRIEREA LISTEI OBIECT FURNIZATE DE COMPILATORUL FORTRAN

Pentru obținerea acestei liste este necesar să specificăm opțiunea OBL în cartela COMPILE FORTRAN.

Lista obiect este alcătuită din trei părți. După cum am menționat, a treia parte a listei conține imaginea instrucțiu-

nilor mașină obținute prin compilarea programului sursă. În acest paragraf ne vom ocupa de primele două părți ale listei.

Prima parte a listei obiect

Aici sînt înscrise informații ale sistemului de gestiune a fișierelor. Această primă parte a listei apare numai în cazul programului principal.

Pentru fiecare fișier utilizabil în program este creată o tabelă de descriere (TDF) și o zonă tampon, ambele implantate în programul principal; de asemenea înaintea programului principal este implantată o zonă tampon numită *zonă tampon FORTRAN*. Toate aceste informații apar în prima parte a listei obiect după cum urmează:

— primul cuvînt conține numărul total N al numerelor de referință FORTRAN atașate fișierelor; acest număr este cel puțin egal cu 3 (corespunzător fișierelor de ieșire prin imprimare, ieșire prin perforare și fișierului de lucrări).

— al doilea cuvînt conține adresa zonei tampon FORTRAN

— următoarele 2N cuvinte corespund celor N numere de referință FORTRAN folosite. Pentru fiecare număr de referință este rezervat un cuvînt dublu în care apar: numărul de referință FORTRAN (4 octeți) și adresa cuvîntului care precede tabela de descriere a fișierului corespunzător. Primele trei cuvinte duble corespund în ordine fișierului de ieșire prin imprimare, fișierului de lucrări și fișierului de ieșire prin perforare (dacă unul dintre aceste fișiere nu este folosit, cuvîntul dublu corespunzător conține FFFFFFFF00000000). De asemenea pentru fiecare din primele trei cuvinte duble al doilea cuvînt este o adresă TDF pentru fișierul sistem corespunzător fiind completată de către editorul de legături.

În continuare, prima parte a listei obiect conține tabelele de descriere a fișierelor folosite în program, pe care le descriem pe scurt în continuare.

Tabelele de descriere a fișierelor ocupă cîte 23 cuvinte; a patra jumătate de octet conține 1,2,3 corespunzător idexurilor *1, *2, *3.

Tabela de descriere a unui fișier pe disc exploatat în acces direct ocupă 88 octeți dintre care menționăm:

— octeții 72—75: adresa zonei tampon
— octeții 76—77: lungimea zonei tampon
— octeții 78—79: lungimea articolului
— octeții 80—81: numărul de articole dintr-o pagină
— octeții 82—83: numărul de sectoare DIAM necesare pentru scrierea unui bloc
— octeții 86—87: numărul de sectoare DIMAS necesare pentru scrierea unui bloc

Tabela de descriere a unui fișier de organizare nedefinită exploatat secvențial ocupă 100 octeți, dintre care octeții 72—87 au aceeași semnificație ca mai sus. În plus menționăm că a 6-a jumătate de octet conține (în binar) 0, 4, 5, 8, 9 după cum articolele sînt de format U, F, FB, V, VB.

Tabela de descriere a unui fișier de organizare secvențială (determinată de prezența opțiunii SFD în DEFINE FILE) ocupă 92 octeți.

A doua parte a listei obiect

Această parte conține în ordine:

— textul instrucțiunilor FORMAT din program (scrise în codul EBCDIC)
— o zonă corespunzătoare instrucțiunilor NAMELIST
— o zonă care conține imaginile variabilelor și tablourilor inițializate prin instrucțiunea DATA
— o zonă conținînd informații referitoare la corespondența între argumente și parametri (la apelarea subprogramelor)
— o zonă în care sînt prezentate în hexazecimal constantele care apar în program

Pentru fiecare instrucțiune NAMELIST, în zona corespunzătoare două cuvinte sînt rezervate pentru a memora numele NAMELIST. Pentru fiecare variabilă NAMELIST sînt rezervate:

— două cuvinte pentru numele variabilei
— 1 cuvînt care conține tipul și adresa variabilei
— 1 cuvînt care conține n (numărul de dimensiuni al variabilei)

Scanned with OKEN Scanner

— n cuvinte care conțin primele n—1 valori maxime ale indicilor și dimensiunea totală

Considerăm următorul program FORTRAN:

```
* DEFINE FILE XO(DVT:MT,RCF:FB,BFS:50,RCS:10)=100
      INTEGER A(100),POZ
      CALL PRIME(A,100,LIM,200,IERR)
      WRITE(100,1) (A(I),I=1,LIM)
    1 FORMAT(I4,6X)
      ENDFILE 100
    2 READ(100,1,END=4) M
      WRITE(108,5) M
      GOTO 2
    4 CALL FIBONACI(A,20,POZ,1200,IERR)
      BACKSPACE 100
      WRITE(100,1) (A(K),K=1,POZ)
    5 FORMAT(' ',I6,3X)
      ENDFILE 100
    6 READ(100,1,END=7) M
      WRITE(108,5) M
      GOTO 6
    7 STOP
      END
```

Vom lista în continuare primele două părți ale listei obiect obținute prin compilarea programului principal din exemplul de mai sus:

```
0000  00000004 000000F4 0000006C 00000000 00000069 60000000
0018  FFFFFFFF 00000000 00000064 00000028
0028  00060500 F4E7F006 44000000 23000100 00000000 00000000
0040  00000000 00000000 00000000 00000000 00000000 0F010032
0058  00000007 00000000 00000000 00000000 00000000 00000000
0070  00000080 0032000A 00010001 00010004
0080  7FFFFFFF 00000000 00000800
017C  4DC9F46B F6E75D40
0184  4D7D407D 6BC9F66B F3E75D40
0338  00000064 00000008 00000014 02000190 02000338 02000324
0350  02000330 02000328 00000014 000004B0 00000014 02000190
0368  02000358 02000320 02000350 02000328
```

Primele două rînduri (octeții 0—27) arată că sînt patru fișiere, din care ultimul nu e de sistem. El are precizată și adresa de implantare a TDF-ului, editorul de legături făcînd acest lucru pentru cele de sistem. Dacă o să consultăm porțiunea corespunzătoare din vidajul partiției în care este rulat programul, vom găsi adresele precizate, iar la adresele respective vom găsi tabela de descriere a fișierului respectiv. Următoarele cinci rînduri (octeții 28—8C) constituie TDF-ul fișierului cu idexul X0. Cele două rînduri care urmează (octeții 17C—18F) sînt listele FORMAT din instrucțiunile 1 și 5 din programul sursă. Octeții 338—377 reprezintă informațiile necesare pentru apelul subprogramelor PRIME și FIBONACI (lista adreselor argumentelor, lista unor constante utilizate, alte informații necesare). Restul octeților constituie zone de lucru sau informații care nu sînt completate de către compilatorul FORTRAN. Conținutul lor este vizibil în vidajul partiției unde este rulat programul.

DETECTAREA INSTRUCȚIUNII DIN PROGRAM CARE A PRODUS EROAREA

Am văzut că acest lucru este posibil numai (în cadrul restrictiv al acestui manual) dacă este vorba de o eroare la execuție din ultimele două categorii.

Pentru determinarea instrucțiunii sînt necesare lista modulelor obiect (care se poate obține specificînd opțiunea OBL în cartela COMPILE) și vidajul memoriei.

Înainte de a apare vidajul propriu-zis al memoriei, apar conținuturile celor 16 registre generale.

Registrul 14 conține adresa de lansare a programului. Pentru a afla adresa de la care este încărcat fiecare modul se adaugă această adresă la adresa relativă de implantare a modulului listată în urma editării legăturilor.

Să ne plasăm deocamdată în cazul în care nu folosim segmentarea programului.

Folosind adresa instrucțiunii în cod mașină care a produs eroarea și adresele reale de implantare în memorie ale modu-

Scanned with OKEN Scanner

lelor (determinate ca mai sus) putem determina modulul care conține instrucțiunea ce a condus la eroare. Scăzînd din adresa instrucțiunii în cod mașină adresa de implantare a acestui modul obținem adresa relativă (la începutul modulului) a instrucțiunii în cod mașină. Cu adresa relativă ne întoarcem la lista modulului obiect. Pe fiecare rînd al acesteia apare la început adresa relativă a instrucțiunii în cod mașină respective. Aceasta ne permite detectarea instrucțiunii în cod mașină care a produs eroarea și în același timp numărul liniei (din unitatea de program care a generat modulul) care conține instrucțiunea FORTRAN care a condus la eroarea semnalată.

Pentru exemplificare să considerăm lucrarea:

```
        JOB TEST,AN:GCUE,PN:HOPE
        COMPILE FORTRAN,OBL,MAP
      COMMON B
      A=1.
      CALL S(A)
      WRITE(108,1)A,B
    1 FORMAT(' ',2F10.3)
      STOP
      END
      SUBROUTINE S(A)
      COMMON B
      C=0.
      B=A/C
      A=A+1
      END
        LINK
        RUN UD:CP
        EOJ
```

La execuție apare mesajul:

ERREUR PROGRAME 00000108

din care deducem că s-a produs o depășire superioară în virgulă mobilă.

Cuvîntul de stare al programului apare listat sub forma:

ETAT PROGRAMME 00006314 42130000

de unde rezultă că adresa din memorie a instrucțiunii în cod mașină care a produs eroarea este 6314.

Registrul 14 conține 000060D0.

Editorul de legătură produce următoarea listă:

```
SEGMENT  F%BLK    NO  1    IMPLANTATION   0
         MODULE   F%BLK    IMPLANTATION  38
         MODULE   F%MDATA  IMPLANTATION  40
         MODULE   S        IMPLANTATION 208
         MODULE   I%INIT   IMPLANTATION 260
         .   .   .   .   .   .   .
```

Adresa instrucțiunii în cod mașină care a produs eroarea, relativă la adresa de lansare a programului este 6314—60D0 = 244. Cercetînd lista produsă de editorul de legături obținem că instrucțiunea se află în modulul S. Relativ la începutul acestui modul, instrucțiunea are adresa 244—208=3C.

Din lista modulului obiect rezultat prin compilarea subrutinei S extragem:

```
LIGNE: 3   0030   0 6 0 0 0A 0018   LD4,0   4[illegible]8
           0034   0 6 0 0 5A 0000   ST4,0   C
LIGNE: 4   0038   1 6 0 0 0A 0014   LD4,0   *A
           003C   0 6 0 0 6F 0000   DVF4,0   C
           0040   0 6 0 0 5A 0000   ST4,0   B
```

Folosind adresa relativă 3C, obținem că instrucțiunea în cod mașină care a produs eroarea corespunde instrucțiunii ASSIRIS:

DVF4,0 C

Instrucțiunea FORTRAN căutată este cea care se găsește pe linia 4 în unitatea de program constituită de subrutina S, deci este următoarea:

B = A/C

Studiind această instrucțiune, obținem și informația suplimentară că eroarea se datorește unei încercări de împărțire cu 0.

Să trecem acum la cazul în care în program se folosește segmentarea. După cum știm, în cazul segmentării segmentele se suprapun prin încărcare și deci adresa instrucțiunii nu mai este suficientă dacă nu știm care este segmentul care este în momentul producerii erorii încărcat în memorie și care conține instrucțiunea respectivă. Deci pentru a putea face calculele de mai sus ne este necesar să știm care dintre segmente se află în memorie.

Precizăm mai întîi că un rînd din vidajul memoriei are următoarea formă (evident, este vorba de un exemplu):

007220 5D404040 4D7D407D 6BF5E76B 7DC96540

avînd următoarea semnificație:

— la adresa 007220 se află cuvîntul 5D404040;
— la adresa 007224 se află cuvîntul 4D7D407D;
— la adresa 007228 se află cuvîntul 6BF5E76B;
— la adresa 00722C se află cuvîntul 7DC96540;

(cuvîntul este precizat prin cele 8 cifre hexazecimale corespunzătoare celor 4 octeți ai săi).

Registrul 15 conține adresa din memorie unde se află *tabela de segmente*. Începînd de la această adresă apar în ordine:

— un cuvînt care conține adresa de unde începe în memorie zona de informații generale (despre care nu vom vorbi aici);
— cîte un cuvînt pentru fiecare segment, în ordinea în care segmentele apar în lista produsă de editorul de legături. Prima cifră hexazecimală este 0 sau 8 (după cum la momentul respectiv de timp segmentul se află sau nu în memorie); iar ultimele 4 cifre hexazecimale indică adresa reală de încărcare în memorie a segmentului.

Vom exemplifica prin descrierea vidajului memoriei *la sfîrșitul* executării programului din capitolul relativ la segmentare.

Registrul 14 conține 000070F0.
Registrul 15 conține 000070C4.

Există 10 segmente. Cuvintele din tabela de segmente referitoare la segmente sînt următoarele (precizăm și semnificația lor folosind și ordinea în care apar segmentele în lista produsă de editorul de legături, listă care a fost dată în capitolul 7):

Adresă în memorie	Cuvînt	Prezență în memorie	Adresă de încărcare	Nume
70C8	000070F0	da	70F0	PRINC
70CC	0000AD18	da	AD18	F%BLK
70D0	8000AD50	nu	AD50	PAR
70D4	0000AD50	da	AD50	MAXIM
70D8	0000ADD8	da	ADD8	MAXIM
70DC	80C0AE00	nu	AE00	OVCT
70E0	0000AE00	da	AE00	A
70E4	0000B020	da	B020	BETA
70E8	8000B080	nu	B080	B
70EC	0000B080	da	B080	C

Observăm că adresele reale de implantare în memorie ale modulelor se pot obține și prin adunarea conținutului registrului 14 la adresele relative de implantare în memorie.

Din tabelul de segmente rezultă că în momentul terminării execuției programului se aflau în memorie segmentele: PRINC, F%BLK, MAXIM, MAXIM, A, BETA, C lucru care rezultă și din analiza programului.

Capitolul XI

PROGRAME UTILITARE

Pentru ca un sistem de operare să fie cît mai eficient, el trebuie să fie însoțit de o mulțime de funcții elementare care să realizeze operațiile cele mai des folosite în gestiunea informațiilor aflate pe suporturi magnetice. Acest rol este îndeplinit de programele utilitare.

Programele utilitare uzuale sînt programele de inventariere a fișierelor, programul de întreținere a fișierelor și programul de generare a fișierelor de încercări. Datorită funcțiilor sale, programul bibliotecar poate fi și el încadrat în programele utilitare.

Programele utilitare lucrează sub controlul monitorului de înlănțuiri într-o partiție serie. Pentru ca execuția lor să necesite cît mai puțin timp, ele sînt catalogate în B.U.S.

În această lucrare ne vom referi numai la programul de întreținere a fișierelor și la programul de inventariere a fișierelor..

PROGRAMUL UTILITAR DE ÎNTREȚINERE (MAINTENANCE)

Acest program utilitar acționează asupra datelor stocate pe suporturi magnetice. Obiectul acțiunii este fie volumul, fie fișierul.

Dacă programul bibliotecar este acela care asigură posibilitatea lucrului cu biblioteci de programe, programul utilitar MAINTENANCE oferă posibilitatea efectuării operațiilor generale de întreținere a fișierelor de date.

Principalele facilități sînt următoarele:

— compararea a două fișiere
— corectarea unui fișier
— duplicarea unui fișier sau a unui volum.

Pentru aducerea în memoria internă a programului utilitar de întreținere și executarea lui, este folosită una dintre cartelele de comandă:

- SYSRUN MAINT
- RUN FN:MAINT

Această cartelă este urmată de cartele parametru care precizează acțiunea pe care trebuie să o întreprindă programul de întreținere și descrierea fișierelor sau volumelor la care se referă această acțiune.

Sintaxa cartelelor parametru este aceeași cu cea a cartelelor de comandă ale bibliotecarului.

Sfîrșitul cartelelor parametru e indicat de cartela % END.

Celelalte cartele parametru pot fi împărțite în două categorii:

— cartelele care precizează fișierele și/sau volumele afectate.
— cartelele care precizează acțiunea care trebuie întreprinsă.

Descrierea fișierelor și volumelor

După cum am menționat, suporturile pot fi numai discuri sau benzi magnetice.

Oricare dintre acțiunile programului utilitar MAINTENANCE se extinde asupra a două fișiere sau a două volume la care ne vom referi sub denumirea de fișierul 1 și fișierul 2, respectiv volumul 1 și volumul 2.

Cu excepția cazului cînd se face comparare, rezultatul acțiunii asupra fișierului 1 (volumul 1) este înscris în fișierul 2 (volumul 2)

Ca regulă generală, suporturile trebuie să fie de aceeași natură (bandă←→bandă sau disc←→disc); în plus trebuie să coincidă și tipul volumului. Precizări vor fi date la fiecare funcție în parte.

1) *cartela OLDFLE* descrie fișierul 1. Forma ei este următoarea:

% OLDFLE DV:*perif*, FN:'*nume*' [,{GN:*g* / CD:*dată*}]

[,VN:*v*] [,UN:*u*]
[,FS:*f*] [,ON:'*proprietar*'] [,FP:'*parolă*']
[,BEG:(*Znz*[,*Bnb*])]

unde parametrii sînt cei cunoscuți de la discuția asupra fișierelor, cu excecpția celor din opțiunea BEG

nz — numărul zonei fișier (numărul zecimal cuprins între 1 și 99)

nb — numărul blocului în zonă (număr zecimal între 1 și 99999)

Parametrii *nz* și *nb* specifică poziția din fișier începînd de la care programul de întreținere să-și înceapă acțiunea. Prin lipsă *nz* = *nb* = 1.

Reamintim că prin zonă fișier înțelegem zona alocată fișierului (pe disc) sau bobina sau fracțiunea de bobină alocată fișierului (pe bandă).

2) *cartela NEWFLE* descrie fișierul 2. Forma ei este următoarea:

% NEWFLE DV:*perif*, FN:'*nume*' [,GN:*g*] [,VN:*v*] [,UN:*u*]

[,FS:*f*] [,ON:'*proprietar*'] [,FP:'*parolă*'] [,AM:{ANY / NEW / OFL}]

[,SZ:*s*][,{OD:*dată* / OG:*g*}] [,OV:*v*] [,BEG:(*Znz*[,*Bnb*])]

unde toți parametrii au fost descriși sau în cartelele de comandă pentru fișiere sau în cartela OLDFLE.

Dacă în NEWFLE nu apare VN : *v*, atunci în fișierul nou creat numărul de versiune al fișierului inițial este mărit cu o unitate.

Scanned with OKEN Scanner

Dacă opțiunea AM lipsește, programul presupune că alocarea a fost făcută de către monitor.

Opțiunile OD, OG, OV se folosesc dacă apare AM : OFL.

3) *cartela OLDVOL* descrie volumul 1. Forma ei este următoarea:

% OLDVOL DV: *perif* [,ON:'*propriclar*'] [,BEG:B*nb*]

unde *nb* este numărul primului bloc afectat

4) *cartela NEWVOL* descrie volumul 2. Forma ei este următoarea:

% NEWVOL DV:*perif* [,ON:*propriclar*'] [,BEG:B*nb*]

Descrierea facilității programului de întreținere

1) Schimbarea suportului de exploatare

Această facilitate constă în duplicarea unui fișier sau a unui volum, noul fișier sau volum (creat pe un nou suport), putînd fi direct exploatabil.

În cazul în care obiectul acțiunii este volumul, acesta poate fi duplicat numai în întregime; suporturile pentru volumul 1 și volumul 2 trebuie să fie de aceeași natură.

La duplicarea fișierelor secvențiale nu există restricții asupra suporturilor.

În cazul fișierelor nedefinite, ambele suporturi trebuie să fie discuri magnetice.

Schimbarea suportului de exploatare este comandată de cartela parametru COPY a cărei formă este următoarea:

$$\%\ \text{COPY}\ \left\{\begin{array}{l}\text{VOL}\\ \text{FLE,}\left\{\begin{array}{l}\text{SEQ}\\ \text{UND}\end{array}\right\}\left\{\begin{array}{l}\text{EOF}\\ Znz\end{array}\right\}\end{array}\right\}$$

Se folosește SEQ sau UND după cum fișierul 1 este secvențial sau nedefinit. Dacă apare EOF, duplicarea se extinde asupra întregului fișier; opțiunea *Znz* limitează la *nz* numărul de zone care trebuie copiate.

2) Protecția fișierelor și volumelor

Această funcție este folosită pentru a stoca provizoriu pe o bandă sau pe un disc de tip diferit, informația aflată pe un disc. Protecția se referă la un volum sau la un fișier nedefinit.

Noul fișier sau noul volum creat nu este direct exploatabil; pentru a putea fi folosit, el trebuie restaurat pe suportul inițial sau pe un suport de aceeași natură.

Pentru a realiza stocarea se folosește cartela parametru SAVE a cărei formă este următoarea:

$$\% \text{ SAVE } \begin{Bmatrix} \text{VOL} \\ \text{FLE} \end{Bmatrix}$$

Pentru verificarea operației de stocare se folosește cartela

$$\% \text{ SCHECK } \begin{Bmatrix} \text{VOL} \\ \text{FLE} \end{Bmatrix}$$

Menționăm că rezultatul stocării este un fișier de organizare nedefinită pe un suport standard.

Pentru restaurarea informațiilor trebuie folosită cartela parametru REST; forma ei este următoarea:

$$\% \text{ REST } \begin{Bmatrix} \text{VOL} \\ \text{FLE} \end{Bmatrix}$$

Pentru verificarea operației de restaurare se folosește cartela:

$$\% \text{ RCHECK } \begin{Bmatrix} \text{VOL} \\ \text{FLE} \end{Bmatrix}$$

3) *Compararea fișierelor sau volumelor*

În cazul în care se compară două volume, ele trebuie să fie de aceeași natură. Dacă este vorba de benzi, compararea se face bloc cu bloc. Dacă este vorba de discuri, compararea se face sector cu sector. Orice neconcordanță este semnalată pe fișierul de ieșire prin imprimare, tipărindu-se :

— numărul blocului sau sectorului volumului 1
— conținutul blocului sau sectorului volumului 1
— numărul blocului sau sectorului volumului 2
— conținutul blocului sau sectorului volumului 2.

Compararea continuă până la detectarea a maximum 10 neconcordanțe.

În cazul comparării fișierelor secvențiale, suporturile pot fi de aceeași natură sau de naturi diferite.

Unitatea de comparare este articolul. Neconcordanțele sînt semnalate prin imprimarea următoarelor informații:

— numerele zonei, blocului de zonă, articolului din bloc al fișierului 1;
— conținutul articolului fișierului 1;
— numerele zonei, blocului de zonă, articolului din bloc al fișierului 2;
— conținutul articolului fișierului 2.

În momentul întîlnirii unei neconcordanțe, programul de întreținere analizează dacă aceasta se datorește unui articol sau bloc lipsă dintr-unul dintre fișiere; în caz afirmativ, se face sincronizarea și comparația continuă. La compararea fișierelor pe disc exploatate în acces direct, unitate de comparare este sectorul, neconcordanțele sînt semnalate prin imprimarea următoarelor informații:

— numerele zonei și sectorului din zonă din fișierul 1;
— conținutul sectorului din fișierul 1;
— numerele zonei și sectorului din zona din fișierul 2;
— conținutul sectorului din fișierul 2.

Funcția de comparare a programului de întreținere este comandată de cartela parametru COMPARE.

Forma acesteia este următoarea:

$$\%\ \text{COMPARE}\left\{\begin{array}{l}\text{VOL, STD,}\left\{\begin{array}{l}\text{EOF}\\ \text{BL}nb\end{array}\right\}\\ \text{FLE,}\left\{\begin{array}{l}\text{SEQ,}\left\{\begin{array}{l}\text{EOF}\\ \text{R}na\end{array}\right\}\\ \text{UND,}\left\{\begin{array}{l}\text{EOF}\\ \text{BL}nb\end{array}\right\}\end{array}\right.\end{array}\right\}$$

STD arată faptul că organizarea este standard.

Compararea se poate face:

— pînă la sfîrșitul volumului sau fișierului (dacă specificăm EOF)

— numai asupra a *na* ($1 \leqslant na \leqslant 99999$) înregistrări (dacă specificăm R*na*)
— numai asupra a *nb* ($1 \leqslant nb \leqslant 99999$) blocuri.

4) *Corectarea unui fișier*

În cazul fișierelor secvențiale, fișierul 2 se obține prin corectarea fișierului 1 în sensul suprimării, adăugării sau modificării (duplicarea, suprimarea, înlocuirea sau adăugarea unui număr de octeți) unui articol. Suporturile celor două fișiere pot fi de aceeași natură sau de naturi diferite.

Trebuie respectată regula conform căreia în cazul articolelor de format fix lungimea articolului să nu se modifice, iar în cazul articolelor de format variabil lungimea articolelor corectate sau adăugate să nu depășească lungimea maximă prevăzută.

Corectarea unui articol determină imprimarea poziției acestuia prin tipărirea numerelor zonei, blocului de zonă și articolului din bloc; de asemenea mai sînt tipărite:
— conținutul articolului vechi (la suprimare)
— conținutul articolului vechi și al articolului nou (la modificare)
— conținutul articolului reper și al articolului nou introdus (la adăugare).

Articolele care se corectează trebuie indicate în ordinea în care se află în fișier.

În cazul fișierelor pe disc exploatate în acces direct, unitatea de corectare este sectorul. Singura corecție care se poate face este modificarea unui sector.

Ca și mai sus, modificarea constă în duplicarea, suprimarea, înlocuirea sau adăugarea unui număr de octeți. La modificare sînt imprimate următoarele informații:
— numărul zonei și blocului din zonă
— conținutul vechiului bloc,
— conținutul noului bloc.

Pentru executarea funcției de corecție trebuie introdusă cartela parametru:

% CORRECT $\left\{\begin{matrix}\text{REC}\\\text{BLK}\end{matrix}\right\}$

(unde REC se pune pentru fișiere secvențiale, iar BLK pentru fișiere pe disc în acces direct), urmată de una sau mai multe cartele DLT, ADD sau RPL care cer respectiv suprimarea unui articol adăugarea unui articol, înlocuirea unui articol sau bloc.

Suprimarea unui articol se face cu ajutorul cartelei:

% DLT (*Znz, Bnb, Rna*)

Adăugarea unui articol se face cu ajutorul cartelei

% ADD (*Znz, Bnb, Rna*)

Modificarea unui articol sau bloc este cerută prin cartela:

% RPL (*Znz, Bnb, Rna*), *listă de opțiuni*

unde lista de opțiuni poate conține următoarele opțiuni (separate între ele prin virgulă):

C*n* — se copiază *n* octeți ($1 \leqslant n \leqslant 999$)

S*n* — se suprimă *n* octeți ($1 \leqslant n \leqslant 999$)

RA*n*:*a* — se înlocuiesc *n* octeți ($1 \leqslant n \leqslant 99$) cu șirul de valori alfanumerice *a*

IA*n*:*a* — se inserează *n* octeți ($1 \leqslant 9 \leqslant 99$) a căror valoare este precizată în șirul de valori alfanumerice *a*.

Ordinea în care trebuie plasate cartelele parametru este următoarea:

— cartelă COPY, SAVE, REST, COMPARE, CORRECT, SCHECK, RCHECK

— cartelă OLDFLE SAU OLDVOL

— cartelă NEWFLE sau NEWVOL

— cartele DLT, ADD, RPL (numai pentru funcția de corecție)

— cartela END

În cadrul aceleiași execuții a programului utilitar MAINTENANCE se pot folosi mai multe funcții ale acestuia.

PROGRAMUL UTILITAR DE INVENTARIERE INVFICH

Acest program dă posibilitatea utilizatorului să obțină informația asupra conținutului unui fișier sau a unui volum.

Volumele tratate pot fi benzi sau discuri magnetice. Ne vom ocupa doar de fișierele care au suport benzi sau discuri magnetice.

Principalele funcții ale programului de inventariere a fișelor sînt următoarele:

— obținerea rezumatului informațiilor aflate pe o bandă sau un disc

— imprimarea conținutului unui fișier sau a unui volum.

Lansarea programului INVFICH se face prin intermediul uneia dintre cartelele de comandă

- SYSRUN INVFICH
- RUN FN: INVFICH

urmată de cartelele parametru în care se precizează la care dintre funcțiile acestui program se face apel precum și informații asupra fișierului sau volumului asupra căruia acționează aceste funcții.

Sintaxa cartelelor parametru este aceeași cu cea a cartelelor bibliotecarului.

Sfîrșitul cartelelor parametru este indicat de cartela % END

Descrierea fișierelor și volumelor

Descrierea fișierelor (cu organizare standard) se face cu ajutorul cartelei parametru DFLE a cărei formă este următoarea:

% DELE DV:*perif*.[FN:'*nume*'[,{GN:*g* / CD:*dată*}][,VN:*n*][,FN:*n*]

[,ON:'*proprietar*'] [,FP:'*parola*'] [,BEG:{(Z*nz*) / ([Z*nz*,]B*nb*)}]

unde semnificațiile opțiunilor sînt aceleași ca și cartela OLDFLE din programul utilitar MAINT.

Opțiunea BEG indică locul de începere a acțiunii programului INVFICH.

Descrierea volumelor se face cu ajutorul cartelei parametru DVOL, a cărei formă este următoarea:

%DVOL DV:*perif* [,ON:'*proprietar*']

cu semnificațiile cunoscute.

Descrierea funcțiilor programului de inventariere a fișierelor

1) *Obținerea rezumatului unui volum*

Se realizează prin introducerea cartelei parametru:

% RESUME

În cazul benzii magnetice, rezumatul constră în listarea tuturor etichetelor de pe bandă, a mărcilor de fișier și a numărului de blocuri consecutive din fiecare fișier, aceste informații apărînd în ordinea în care ele se află pe bandă. Rezumatul se oprește în întîlnirea a două mărci de fișier consecutive.

În cazul discurilor este listată zona informațiilor de control, aflată, după cum știm, pe cilindrul 0. Forma în care apar aceste informații va rezulta din exemplul atașat la sfîrșitul capitolului.

2) *Obținerea conținutului unui fișier*

Această funcție a programului de inventariere a fișierelor este comandată prin cartela parametru DUMP, a cărei formă este următoarea:

$$\%\ \text{DUMP}\left\{\begin{array}{l}\text{SEQ,}\left\{\begin{array}{l}\text{EOF}\\ \text{R}na\end{array}\right\}\\ \text{UND,}\left\{\begin{array}{l}\text{EOF}\\ \text{BL}nb\end{array}\right\}\end{array}\right\}$$

Opțiunea EOF arată că listarea conținutului se face pînă la sfîrșitul fișierului.

Dacă folosim R*na*, vor fi listate numai *na* articole.

Dacă folosim BL*nb*, vor fi listate numai *nb* blocuri.

Forma în care apar informațiile din fișier va fi descrisă pe exemplul atașat la sfîrșitul capitolului. Cu titlu informativ menționăm că există și posibilitatea alegerii de către utilizator a formei în care să apară listarea informațiilor din fișier.

Ordinea în care trebuie plasate cartelele parametru este următoarea:

— cartela RESUME sau cartela DUMP
— cartela DVOL sau cartela DFLE
— cartela END

cu mențiunea că în cadrul aceleiași lansări a programului INVFICH pot fi efectuate mai multe operații de inventariere.

Exemplu

Exemplul constă din trei lucrări.

Prima lucrare este următoarea:

```
JOB CREARE,AN:1218,PN:HOPE
INIT DV:RD1,VS:111111
ALLOC DV:RD1,FN:'SECV',VN:1,GN:1,AM:ANY,SZ:1
ASSIGN S,DV:RD1
LABEL S,FN:'SECV',VN:1,GN:1
ALLOC DV:RD1,FN:'NEDEF',VN:1,GN:1,AM:ANY,SZ:1
ASSIGN N,DV:RD1
LABEL N,FN:'NEDEF',VN:1,GN:1
INIT DV:MT1,VS:AAA111
ASSIGN T,DV:MT1
LABEL T,FN:'TAPE',
INIT DV:RD2,VS:111222
ALLOC DV:RD2,FN:'F2',AM:ANY,SZ:1,VN:1,GN:1
ALLOC DV:RD1,FN:'BIBLIA SOU',VN:1,GN:1,AM:ANY,SZ:1
ALLOC DV:RD1,FN:'BIBLIA REN',VN:1,GN:1,SZ:1,AM:ANY
ALLOC DV:RD1,FN:'BIBLIA IER',GN:1,VN:1,AM:ANY,SZ:3
SAVNU BIBLIO
```

```
% OPNLIB S,DV:RD1,VN:1,GN:1,FT:SOU,LN:BIBLS,INIT
% OPNLIB C,DV:RD1,VN:1,GN:1,FT:RBN,LN:BIBLC,INIT
% OPNLIB L,DV:RD1,VN:1,GN:1,FT:INT,LN:BIBLL,INIT
% INCLUDE OL:*1,DL:S,OF:S1
% ENDLIB
.          COMPILE FORTRAN,DV:RD1,LN:BIBLS,VN:1,GN:1,FN:C1
* FETCHS LN:BIBLS,DV:RD1,VN:1,GN:1,FN:S1
* DEFINE FILE S2(DVT:RD,RCF:FB,RCS:12,BFS:44,SFD)=1
* DEFINE FILE NO(DVT:RD,RCF:D,RCS:12)=2
* DEFINE FILE TO(DVT:MT,RCF:FB,RCS:12,BFS:36)=3
      DIMENSION NUME(2)
    8 READ(105,1,END=9)NUME,IV,I
    1 FORMAT(3A4,I2)
      WRITE(1,4)NUME,IV
      WRITE(2'I,4)NUME,IV
      WRITE(3,4)NUME,IV
      GOTO 8
    4 FORMAT(3A4)
    9 STOP
      END

% ENDLIB
.          COMPILE FORTRAN,DV:RD1,LN:BIBLC,VN:1,GN:1,FN:C1
* FETCHS LN:BIBLS,DV:RD1,VN:1,GN:1,FN:S1
.          LINK DV:RD1,LN:BIBLL,FN:L1,VN:1,GN:1
.          RUN
POPESCU    74 2
VASILE     80 7
BARDOT     4312
CARDINAL   16 6
LOREN      40 3
POPOVICI   15 1
QUINN      7010
SAMANTHA    0 9
.EOF
.          EOJ
```

Această lucrare creează următoarele fișiere:

Nume	*Idex*	*Periferic*	*Organizare*	*Acces*
SECV	S	RD1	secvențială	secvențial
NEDEF	N	RD1	nedefinită FORTRAN	direct
TAPE	T	MT1	nedefinită FORTRAN	secvențial

De asemenea sînt catalogate în biblioteci aflate pe discul RD1 cărțile:

Numele cărții	Idexul bibliotecii	Numele bibliotecii	Tip
S1	S	BIBLS	sursă
C1	C	BIBLC	B.T.
L1	L	BIBLL	I.M.T.

Cele trei cărți corespund fazelor prin care trece programul FORTRAN care este rulat în cadrul aceleiași lucrări și care realizează scrierea în fișierele menționate mai sus. Fișierele conțin articole de format fix blocat (cîte 3 articole în fiecare bloc); fiecare articol memorează numele și vîrsta unei persoane.

A doua lucrare, al cărei nume este INVENTAR, constituie un model de aplicare a programului utilitar INVFICH. Sînt listate rezumatele volumelor aflate pe unitatea RD1, pe unitatea MT1 precum și conținutul fișierelor create în lucrarea precedentă.

Lucrarea începe cu rezumatul discului aflat pe unitatea RD1. După o zonă rezervată de sistem apare lista etichetelor aflate în zona informațiilor de control precum și tabela de ocupare a volumelor (TOV). Conținutul etichetelor a fost descris în capitolul 4. Cartea BIBREP conține informații asupra caracteristicilor bibliotecilor respective.

În rezumatul benzii aflate pe unitatea MT1 apar etichetele VOL1, HDR1, HDR2, marca de fișier (FM), numărul blocurilor din fișier și etichetele EOF1, EOF2. Structura este monofișier-monovolum.

Urmează listarea conținutului fișierului organizat secvențial pe disc. Apar în ordine numărul zonei, numărul blocului, numărul articolului și conținutul articolului (în scriere hexazecimală și alfanumerică).

```
*          JOB INVENTAR,AN:1210,FN:HOPE
*          SYSTEM INVFIC
INVFICH STARTED
                                    PROGRAMME D'INVENTAIRE FICHIERS VERSION 15
% RESUME
% DVOL DV:RD1
SYS AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAA
VOL1 111111
TOV 1111111100000000000 0000000000000000000 0000000000000000000 0000000000000000000 0000000000000000000
    0000000000000000000 0000000000000000000 0000000000000000000 0000000000000000000 0000000000000000000
HDR1 SBOV                    01 0001 01 77343 77343   000 F 01 001 000120 000239 000 000000 000000
        HDR2  C 00044 F 00012 000000 000003 008
HDR1 [illegible]                01 0001 01 77343 77343   000 F 01 001 000240 000359 000 000000 000000
        HDR2  C 00256
HDR1 BIBLS   [illegible]          01 0001 01 77343 77343   000 F 01 001 000360 000479 000 000000 000000
        HDR2  F 00256 000000 000037 000
                HDR3  BIBREP            000 000001 000002 F 00248
                HDR3  S1                001 000003 000002 F 00080
HDR1 BIBLS   [illegible]          01 0001 01 77343 77343   000 F 01 001 000480 000599 000 000000 000000
        HDR2  F 00256 000000 000038 000
                HDR3  BIBREP            000 000001 000002 F 00248
                HDR3  C1                001 000003 000010 F 00002
HDR1 BIBLL   [illegible]          01 0001 01 77343 77343   000 F 01 001 000600 000959 000 000000 000000
        HDR2  F 01024 000000 000039 000
                HDR3  BIBREP            000 000004 000009 F 01016
                HDR3  L1                001 000012 000112 F 01016
                                    PROGRAMME D'INVENTAIRE FICHIERS VERSION 15
% RESUME
% DVOL DV:RD1
```

```
VOL1 AAA111
HDR1 TAPE                    0001 0001 0001 00  77343  77343  000000
HDR2 U 00036 00000                                  00
FM
NS00003
FM
ZOF1 TAPE                    0001 0001 0001 00  77343  77343  000003
ZOF2 F 00036 00000                                  00
FM
FM
```

PROGRAMME D'INVENTAIRE FICHIERS VERSION 15

```
% DUMP SEQ,EOF
% DPLE DV:RD1,FN:'SEQV',GN:1,VN:1
Z01B00001R001        D7D6D7C5E2C3E4404040 F7F4
                     P O P E S C U        7 4
Z01B00001R002        E5C1E2C9D3C540404040 F8F0
                     V A S I L E          8 0
Z01B00001R003        C2C1D9C4D6E340404040 F4F3
                     B A R D O T          4 3
Z01B00002R001        C3C1D9C4C9D5C1D34040 F1F6
                     C A R D I N A L      1 6
Z01B00002R002        D3D6D9C5D54040404040 F4F0
                     L O R E N            4 0
Z01B00002R003        D7D6D7D6E5C9C3C94040 F1F5
                     P O P O V I C I      1 5
Z01B00003R001        D3E4C9D5D54040404040 F7F0
                     Q U I N E            7 0
Z01B00003R002        E2C1D4C1D5E3C8014040 40F0
                     S A M A N T H A        0
```

PROGRAMME D'INVENTAIRE FICHIERS VERSION 15

```
% DUMP DNP,DN1
% DPLE DV:RD1,FN:'NEDEP',GN:1,VN:1
Z01B00001            D7D6D7D6E5C9C3C94040 F1F5D7D6D7C5E2C3E440 4040F7F4D3D6D9C5D540 40404040F4F0F1F64005
                     P O P O V I C I      1 5 P O P E S C U      7 4 L O R E N        4 0 1 6  0
D9E5D6404040404040C3E2 E360C3C360D7C3C3E6D3 C3C1D9C4C9D5C1D34040 F1F6E5C1E2C9D3C54040 4040F8F0C540E2E3C1D5
R W O                C S  P - C - P - C O L  C A R D I N A L      1 6 V A S I L E        8 0 E  S T A N
```

```
C4C1D9C440C3E8C1D4C1 D5E3C8C1404040F0C6E4 C9D5D54040404040F7F0 C2E3C0C3C3C0D7C0C3C4 D3C0C2C1D9C4C0C34040
D A R D   C S A M A N T H A       Q U I N N          7 0 S T - C O - T - C O   D - P A R D O T
4040F4F3F2C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0
   4 3 2
C0C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0C0C0C0

C0C0C0C0C0C0C0C0C0C0C0 C0C0C0C0C0C0C0
```

PROGRAMME D'INVENTAIRE FICHIERS VERSION 15

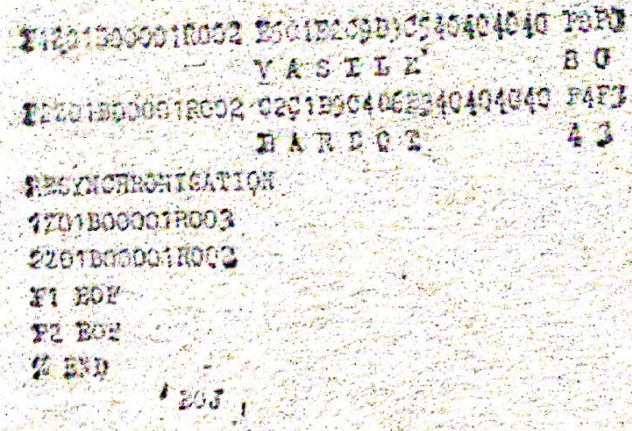
```
% DUMP UND,BL10
% DFLS DV:MT1,FN:'TAPE'
ZO1B00001.          D7D6D7C5E2C3E4404040 F7F4E5C1E2C9D3C54040 4040F8F0C2C1D9C4D6E3 4040404040F3
                    P O P E S C U        7 4 V A S I L E          8 0 B A R D O T          4 3
ZO1B00002           C3C1D9C4C9D5C1D34040 F1F6D3D6D9C5D5404040 4040F4F0D7D6D7D6E5C9 C3C94040F1F5
                    C A R D I N A L      1 6 L O R E N            4 0 P O P O V I C I      1 5
ZO1B00003           D8E4C9D5D54040404040 F7F0E2C1D4C1D5E3C8C1 404040F0
                    Q U I N N.           7 0 S A M A N T H A      0
% END
        EOJ


        JOB MOD,AN:1218,PN:HOPE
        SYSRUN MAINT   : -- T --
MAINT   STARTED
```

PROGRAMME DE MAINTENANCE FICHIERS VERSION 15

```
% CORRECT REC
% OLDFLE DV:RD1,FN:'SECV',GN:1,VN:1
% NEWFLE DV:RD2,FN:'P2',GN:1,VN:1
% SLT (S1,B1,R2)
F1Z01B00001R002 E5C1E2C9D3C540404040 F8F0
                V A S I L E          8 0
```

PROGRAMME DE MAINTENANCE FICHIERS VERSION 15

```
% COMPARE FLE,SEQ,ZOF
% OLDFLE DV:RD1,FN:'SECV',GN:1,VN:1
% NEWFLE DV:RD2,FN:'P2',VN:1,GN:1
```



În listarea conținutului fișierului de organizare nedefinită exploatat în acces direct sînt tipărite următoarele informații asupra blocurilor:

— numărul zonei
— numărul sectorului
— conținutul sectorului (în scriere hexazecimală și alfanumerică)

Cea de a doua lucrare se încheie cu listarea conținutului fișierului de organizare nedefinită de pe bandă. Pentru fiecare bloc sînt listate:

— numărul zonei
— numărul blocului
— conținutul articolelor din bloc (în scriere hexazecimală și alfanumerică)

A treia lucrare, cu numele MOD, constă dintr-o modificare a fișierului secvențial de pe disc (în sensul suprimării unui articol) și în compararea fișierului inițial cu cel rezultat prin modificare.

În urma executării funcției de corectare apare numărul fișierului, numărul zonei, numărul blocului, numărul articolului și conținutul articolului care este suprimat.

La compararea celor două fișiere apar informațiile despre articolele din fișiere care nu coincid (în același format ca la corectare) și apoi sînt listate informațiile despre articolele care sînt puse în corespondență în cadrul încercării de resincronizare a informațiilor din cele două fișiere (resincronizare reușită în cazul nostru).

Capitolul XII

PROBLEME ACTUALE ALE SISTEMELOR DE OPERARE

SISTEME DE CALCUL

Eficiența utilizării unui calculator depinde atît de performanțele tehnice ale acestuia cît și de ansamblul programelor care permit valorificarea acestor performanțe. De aceea prin *sistem de calcul* se înțelege în mod curent ansamblul hardware (echipament) + software (programare) care asigură prelucrarea datelor de intrare conform unor cerințe impuse de utilizatori.

După cum știm, legătura utilizatori — sistem o constituie *lucrarea* (*job* în engleză, *travail* în franceză) care constă din ansamblul datelor de intrare, procedurilor de prelucrare a acestora, resurselor utilizate etc.

Fiecare lucrare consumă *resurse* ale sistemului, cele mai obișnuite fiind următoarele:

— resurse hardware (procesor, memorie etc.)
— resurse software (fișiere, compilatoare, biblioteci etc)

Dacă vom numi *proces* un program în execuție, atunci putem defini un *sistem de calcul* ca fiind o mulțime de funcții de supraveghere și control a proceselor create de utilizatorii săi. Aceste funcții sînt următoarele:

— crearea și înlocuirea proceselor
— controlul evoluției unui proces (în sensul că fiecare proces din sistem evoluează normal și nu blochează continuu evoluția altor procese).
— luarea unor decizii în cazul în care în timpul execuției unui proces apar condiții excepționale (erori aritmetice, erori mașină, întreruperi, violarea împărțirii memoriei etc.)
— alocarea resurselor hardware între procese
— posibilitatea accesului la resursele software

— asigurarea protejării și securității informațiilor prin controlul accesului la ele
— asigurarea comunicării între procese atunci cînd este cazul

Aceste funcții trebuie să fie asigurate de cele două componente ale sistemului de calcul (hardware și software) deoarece procesele nu le pot asigura ele însele decît cu prețul unor mari eforturi de programare.

Funcțiile asigurate de hardware sînt simple, constituind doar baza pentru realizarea funcțiilor software de tipul celor de mai sus, care sînt mult mai complexe. De aceea partea de programare prelungește posibilitățile calculatorului propriu-zis. Sistemul de programe care pe baza hardware-ului existent realizează funcțiile de mai sus este numit de obicei sistem de operare.

În aceste condiții utilizatorul privește sistemul de operare ca un auxiliar care-i asigură legătura cu partea de echipament a calculatorului.

Astfel sistemul de operare este una dintre componentele caracteristice importante ale sistemelor de calcul moderne. Prin existența sa, sistemul de operare permite folosirea eficientă a calculatorului de către utilizatori din cele mai diverse domenii.

În general, hardware-ul unui calculator nu suferă modificări într-un timp scurt, deci poate fi privit ca o componentă statică a sistemului de calcul. Folosind același hardware pot fi create mai multe sisteme de operare care să asigure în mod diferit funcțiile de supraveghere și control cerute de un sistem de calcul. Avînd în vedere că realizarea unui sistem de operare este costisitoare (de exemplu pentru sistemul de operare Disk Operating System al firmei I.B.M. utilizat pe seria de calculatoare 360 a fost necesară o muncă de proiectare și programare echivalentă cu munca a 1000 de oameni în decursul unui an), pentru un anumit tip de calculator se concepe de la început un sistem de operare care să-i folosească cît mai bine posibilitățile tehnice; ulterior, păstrînd în mare structura (concepția) inițială, sistemului de operare i se aduc mereu îmbunătățiri. Astfel pentru seria de calculatoare

FELIX C-256 (IRIS 50) a fost conceput sistemul SIRIS, care a fost apoi continuu îmbunătățit (în momentul de față s-a ajuns la versiunea 15 din SIRIS 3), fără ca ideile de bază ale acestuia să se schimbe.

CARACTERISTICILE UNUI SISTEM DE OPERARE

Astăzi există în lume o mare varietate de sisteme de operare care lucrează pe diferite sisteme de calcul. Lipsa unei teorii formale bine închegate a sistemelor de operare ca și a unei terminologii standardizate face dificilă compararea posibilităților pe care le oferă acestea. Totuși în practică s-au cristalizat cîteva criterii de apreciere a unui sistem de operare; un astfel de criteriu îl constituie modul în care acesta permite utilizarea calculatorului de către cei ce-l folosesc. Vorbind despre acest criteriu, putem delimita mai multe caracteristici pe care trebuie să le avem în vedere:

a) — *Accesul.* Prin aceasta înțelegem posibilitatea pe care o are utilizatorul de a conversa cu sistemul. Astfel putem deosebi sisteme în care este posibilă utilizarea conversațională cu ajutorul terminalelor sau sisteme în care nu este posibilă decît executarea în serie a lucrărilor etc.

b) — *Utilizarea.* În momentul cînd un utilizator are acces la sistem, el poate folosi numai anumite facilități (de exemplu numai un limbaj de programare specific) sau îi este permisă folosirea tuturor posibilităților sistemului de calcul. De asemenea prezintă importanță gradul de ușurință cu care se pot pregăti și structura lucrările.

c) — *Performanța* sistemului este caracterizată prin calitatea serviciilor pe care le poate asigura utilizatorului. Aici sînt incluse probleme legate de securitatea seturilor de date, timpul care se scurge între cererea utilizatorului și satisfacerea ei etc.

Alte caracteristici sînt modul cum memoria principală și memoriile externe sînt alocate utilizatorului, tipul de pre-

Scanned with OKEN Scanner

lucrare al proceselor care sînt în curs de execuție (procesul este executat pentru o perioadă fixă de timp după care i se ia controlul sau procesul se execută pînă la apariția unei condiții excepționale cînd i se ia controlul, etc.), care sînt posibilitățile de a folosi aceleași fișiere de către mai multe procese, dacă configurația sistemului este modulară (în sensul că unei configurații date i se pot aduce modificări prin adăugarea de noi unități) etc.

TIPURI DE SISTEME DE OPERARE

Sistemul de operare fiind un set de programe de control și de prelucrare desemnate să maximizeze utilizarea resurselor de care dispune calculatorul, este normal ca, în funcție de modul în care se rezolvă această problemă, să avem mai multe tipuri de sisteme.

Vom enumera în continuare unele caracteristici definitorii ale sistemelor de operare din punctul de vedere al manipulării lucrărilor.

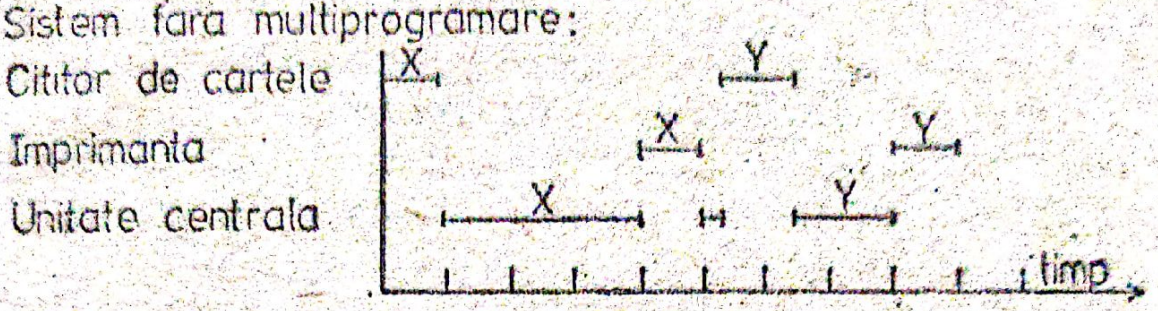
a) — *Sisteme de operare cu multiprogramare*

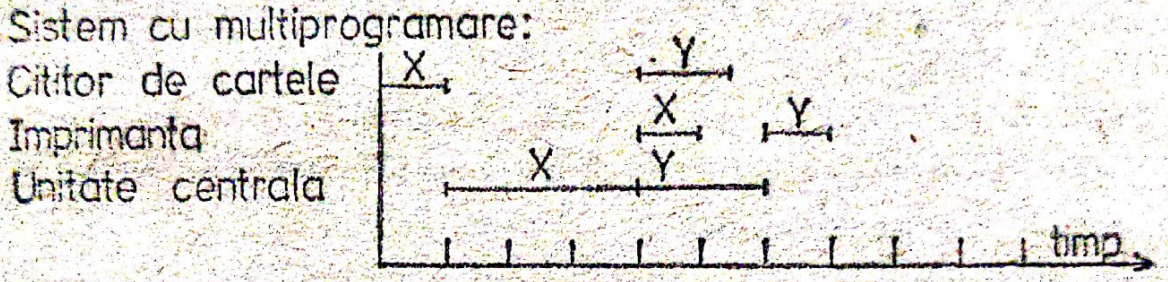
Într-un sistem cu multiprogramare fiecărei lucrări care va fi executată i se atașează o prioritate internă care depinde de prioritatea externă (cea fixată de către dispecerat) și de momentul în care sistemul de operare a luat cunoștință de lucrarea respectivă.

Prioritatea internă a lucrărilor determină acea lucrare care trebuie să fie executată. Astfel lucrarea cu cea mai mare prioritate internă va fi executată pînă cînd apare o condiție în care controlul îi este luată (o cerere de executare a unei operații de intrare/ieșire, terminarea prelucrării etc.). În această situație, programul cu cea mai mare prioritate este lansat; în momentul în care condiția care a determinat luarea controlului programului de prioritate mai mare a fost rezolvată, este reluată eventual execuția acestui program.

Scanned with OKEN Scanner

În cele două diagrame care urmează va apare clar economia de timp care se realizează prin folosirea multiprogramării.





După cum am arătat în capitolul 2, în cazul sistemelor cu multiprogramare, memoria internă este împărțită în mai multe partiții, în fiecare dintre ele fiind prezentă cîte o lucrare. În funcție de prioritatea atașată partițiilor, de prioritatea internă a lucrării și de resursele pe care le folosește, la fiecare moment de timp instrucțiunile unei singure lucrări sînt în curs de execuție.

În general, în sistemele cu multiprogramare accesul utilizatorului se face prin intermediul cititorului de cartele, iar obținerea rezultatelor se face la imprimantă. Prelucrările se fac prin executarea unui lanț de lucrări într-o partiție. Principalul obiectiv urmărit este funcționarea continuă a unității centrale (de unde rezultă implicit un timp mai scurt în obținerea rezultatelor prelucrării).

b) *Sisteme de operare cu acces multiplu* (Time-Sharing Systems)

În aceste sisteme accesul utilizatorului se face în general de la un terminal (mașină de scris, unitate de afișare cu tub catodic etc.). Alocarea memoriei și a timpului de utilizare a sistemului se face pe baze dinamice. La fiecare moment de timp, lucrările utilizatorilor se pot afla fie în memorie (așteptînd să fie executate), fie pe o unitate de memorie auxi-

liară. Controlul este dat pe rînd acestor lucrări pentru un interval fixat de timp. După expirarea acestui interval de timp sau la apariția unei condiții excepționale controlul este trecut altei lucrări.

Trecerea unei lucrări din memoria auxiliară în cea principală și invers se face automat de către sistem pe baza unor criterii dinainte stabilite. Utilizatorul are impresia, datorită timpului scurt în care răspunsul la cererea sa este dat, că este executată numai lucrarea sa.

Un asemenea sistem de operare este indicat pentru învățămînt deoarece dialogul cu sistemul este direct. Utilizatorul poate să-și corecteze programele direct de la terminal și să primească rezultatele prelucrării la acest terminal sau la alte unități de ieșire mai rapide ale calculatorului.

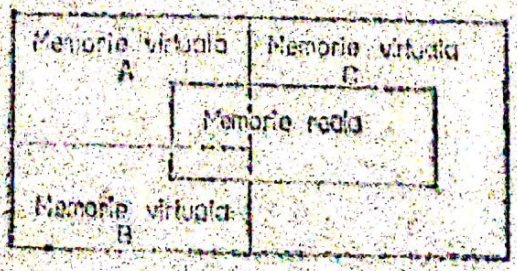
c) — *Sisteme virtuale*

Aceste sisteme pun la dispoziția utilizatorului resurse logice care nu au în mod obligatoriu corespondență fizică în sistem.

Principala caracteristică o constituie *memoria virtuală*. Utilizatorul folosește în programele lui numai memoria internă, deși sistemul utilizează și memorii auxiliare. Aceasta este posibil datorită faptului că la fiecare moment de timp memoria internă conține numai o parte a programului utilizatorului (din care face parte bineînțeles instrucțiunea în curs de executare). Memoria internă și programul sînt împărțite în *pagini*; o pagină care conține o parte a programului este adusă în memoria internă numai atunci cînd este necesar.

Memoria virtuală are un număr mai mare de pagini decît cea reală.

Schema care urmează exemplifică acest lucru pentru un sistem cu multiprogramare folosind memorie virtuală, pentru cazul a trei utilizatori A, B și C:



Transferul paginilor unui program în memoria internă făcîndu-se automat de către sistem, utilizatorul are impresia că folosește un calculator cu o memorie mult mai mare decît cea reală. Astfel el poate rula programe de dimensiuni foarte mari și care prelucrează un volum mare de date.

Generalizînd la nivelul întregului sistem obținem conceptul de *sistem virtual*.

Într-un sistem virtual utilizatorul are posibilitatea de a-și alege sistemul de operare potrivit pentru lucrare. Astfel, exemplificînd pentru sistemul I.B.M., în același moment de timp un utilizator poate să-și execute programul cu ajutorul sistemului D.O.S., pe cînd altul își poate rula programul sub controlul sistemului O.S. etc.

Facem observația că un sistem poate avea în același timp mai multe dintre caracteristicile prezentate mai sus.

Din cele de mai sus rezultă faptul că, în paralel cu îmbunătățirea și creșterea complexității hardwareului unui calculator, are loc și procesul de perfecționare continuă a părții de programare. Aici intervine rolul matematicianului care trebuie să participe la crearea fundamentelor sistemelor de operare și crearea unor sisteme de operare perfecționate pe această bază.

Analizînd sistemul de operare SIRIS din punctul de vedere al celor discutate în acest capitol, observăm că:

— Sistemul SIRIS permite multiprogramarea. Împărțirea memoriei în regiuni este statică (nu permite accesul multiplu și nici nu folosește memorie virtuală).

— În marea lor majoritate, sistemele FELIX actuale nu utilizează modul conversațional de lucru. Se preconizează ca acest mod de lucru să capete o extindere mai mare în viitor.

— În momentul cînd utilizatorul are acces la sistem, el poate utiliza toate programele puse la dispoziție de acesta folosind limbajul de comandă.

— Ca și în alte sisteme de operare, în sistemul SIRIS se caută să se mărească timpul în care unitatea de comandă lucrează, să facă munca operatorului mai ușoară prin diferite

tehnici de lucru. Astfel au fost evidențiate rolurile fișierelor de lucrări, ale tehnicii SYMBIONT și TELESYMBIONT etc.

— O altă facilitate apărută în sistemul de operare SIRIS 3 constă în recunoașterea automată a volumelor. Sistemul își organizează un tabel care conține informații despre toate volumele montate. O lucrare poate fi lansată numai dacă acest tabel arată că toate volumele solicitate de lucrare sînt disponibile. În acest mod se evită pierderea de timp prin începerea executării și întreruperea ei ulterioară la solicitarea unui volum care nu este disponibil dintr-o cauză sau alta.

DIRECȚII DE DEZVOLTARE A TEHNICILOR DE PROGRAMARE

• Performanțele sistemelor de operare sînt strîns legate de existența unor limbaje de programare puternice (și implicit compilatoare eficiente pentru acestea).

Dintre numeroasele limbaje existente în prezent sînt folosite mai des limbajele FORTRAN, COBOL și PL/I. Această situație a furnizat o serie de argumente pentru școala universalistă, care susține crearea unui singur limbaj de programare universal. Plecînd de la această idee au fost create PL/I și ALGOL 68. Dezavantajele (fiindcă avantajele sînt evidente) nu sînt neglijabile, fiind determinate în special de costurile ridicate ale compilatoarelor respective, timpul lung necesar alcătuirii acestora și timpul relativ mare necesar compilării unui program.

O altă tendință în dezvoltarea limbajelor de programare vizează libertatea programatorului de a-și crea limbajul în care dorește să-și scrie programele. Aceasta este, bineînțeles, condiționată de:

— construcția compilatorului respectiv să fie simplă și să poată fi automatizată pe baza unor indicații despre limbaj, date de utilizator;

— utilizatorul să aibă la dispoziție tehnici simple și bine puse la punct pentru a identifica instrucțiunile pe care le implementează în limbajul său.

Scanned with OKEN Scanner

Perspectiva de a avea un limbaj în care problema concretă să se exprime direct, aproape ca în limbajul obișnuit, este ademenitoare. Acest lucru se poate realiza plecînd de la un *limbaj de bază* și adăugînd acele operații sau tipuri de date pe care dorim să le folosim în programul concret. Translatorul va fi realizat automat și va traduce programul nostru în limbajul mașinii. De acest tip de probleme se ocupă *limbajele extensibile*.

• Concomitent cu aceste încercări de a perfecționa limbajele de programare, se lucrează tot mai mult în direcția punerii pe baze științifice, riguroase, a muncii programatorului. Necesitatea unor metodologii de scriere și urmărire a programelor apare mai ales pentru proiectele complexe în care numărul instrucțiunilor scrise este de ordinul miilor sau zecilor de mii. Se realizează astfel economii datorită garanției că în final programul va îndeplini prelucrarea dorită. În literatura de specialitate sînt deseori amintite situații în care echipe de analiști și programatori au lucrat în zadar un timp îndelungat deoarece sistemul de programe obținut nu corespundea scopului fixat. Acest lucru este cu atît mai stringent cu cît un programator poate scrie într-o zi pentru proiectele mari în medie numai 5—10 instrucțiuni corecte.

Metodele create permit urmărirea structurii sistemului de programe și garantează că acesta va corespunde obiectivului fixat. În plus scrierea programelor, depanarea lor și modificarea lor ulterioară sînt mai ușor de realizat. Importanța acestor metode rezultă și din următoarea situație: după o perioadă de timp (de exemplu un an) pentru 90% din programele scrise de către utilizator codificarea din nou a acestora este mai ușoară decît modificarea lor.

Una dintre aceste metode este *programarea structurată*. Folosită împreună cu alte tehnici, randamentul unui programator crește de circa zece ori.

• Orice limbaj de programare, chiar cele de nivel foarte înalt (numite și universale sau algoritmice), este legat de un anumit tip de calculator. Se regăsesc în el structuri de date corespunzătoare calculatorului respectiv: adrese, exe-

cuție secvențială a programului, codificări binare ale informației.

Începînd cu anul 1973 au început să fie făcute cercetări pentru crearea *limbajelor de nivel foarte înalt* (LNFI), apropiate cît mai mult de modul de gîndire al programatorului, care să nu mai fie condiționate de caracteristicile unui tip sau altul de calculator. În aceste limbaje vor fi eliminate restricțiile de execuție secvențială, diferitele grupuri de instrucțiuni putînd fi scrise în mod arbitrar; vor putea fi introduse elemente de programare nedeterministe (toate limbajele de pînă acum permit doar o programare deterministă). De asemenea se va putea opera direct cu mulțimi, obiectele mulțimii fiind regăsite nu după adresă, ci după caracteristicile lor și relațiile cu alte obiecte.

O lucrare într-un asemenea limbaj se va desfășura astfel: se introduc datele și relațiile între ele și se stabilește dialogul cu mașina. Aceasta va organiza datele introduse și, pe baza unor deducții și a unor informații suplimentare, va răspunde cererilor utilizatorului.

Scanned with OKEN Scanner

ANEXA — 1

CODUL EBCDIC

În această anexă vom da pentru fiecare caracter (folosit în FORTRAN sau în cartelele de comandă) reprezentarea sa ca octet (indicînd două cifre hexazecimale) și scrierea sa pe cartele indicînd pozițiile în care se fac perforațiile care îl identifică.

A	C1	12—1	0	F0	0
B	C2	12—2	1	F1	1
C	C3	12—3	2	F2	2
D	C4	12—4	3	F3	3
E	C5	12—5	4	F4	4
F	C6	12—6	5	F5	5
G	C7	12—7	6	F6	6
H	C8	12—8	7	F7	7
I	C9	12—9	8	F8	8
J	D1	11—1	9	F9	9
K	D2	11—2	blanc	40	
L	D3	11—3	.	4B	12—8—3
M	D4	11—4	(	4D	12—8—5
N	D5	11—5	+	4E	12—8—6
O	D6	11—6	*	5C	11—8—4
P	D7	11—7	)	5D	11—8—5
Q	D8	11—8	/	61	0—1
R	D9	11—9	,	6B	0—8—3

Scanned with OKEN Scanner

S	E2	0—2	%	6C	0—8—4
T	E3	0—3	—	6D	0—8—5
U	E4	0—4	;	7A	8—2
V	E5	0—5	'	7D	8—5
W	E6	0—6	=	7E	8—6
X	E7	0—7			
Y	E8	0—8			
Z	E9	0—9			

Scanned with OKEN Scanner

ANEXA — 2

PROGRAM ASSIRIS PENTRU DETERMINAREA TIMPULUI

De multe ori este important să cunoaștem intervalul de timp scurs între două evenimente. De aceea vom plasa înaintea cartelei LINK următoarea secvență de cartele:

```
                ENTSG TIMP
•               COMPILE ASSIRIS
•
TIMP            CSECT
                LD,14       %
                STM,1       SAVE
                GTS
                LD4,1       *2*4
                SRL4,0      1
                ST4,0       *1*4
                LDM,1       SAVE
                BRU         *8*4
SAVE            RES,4       2
                END
```

Aceasta reprezintă o subrutină care este apelată din cadrul unui program FORTRAN printr-o instrucțiune de forma:

CALL TIMP(i)

unde i este o variabilă întreagă care va primi ca valoare timpul unității de comandă măsurat de la începutul lucrării; unitatea de măsură este 1,024 ms.

Dacă dorim o unitate de măsură mai mare, putem modifica instrucțiunea 5, scriind-o sub forma:

SRL4,0 k

unde k este un număr natural. Atunci unitatea de măsură va fi $1{,}024*2^{k-1}$ ms.

Scanned with OKEN Scanner

ANEXA — 3

În această anexă sînt prezentate cele mai frecvent întîlnite incidente detectate de S.G.F. (E.P. semnifică eroare program)

ERORILE DETECTATE DE FUNCȚIILE S.G.F. — MONITOR

Nr. de eroare	*Mesaj*	*Natura erorii*
00	SEG. INCONNU	E.P. sau memorie central distrusă
01	OUV. F. INCONNU	Încercare de deschidere a unui fișier deja deschis (E.P.)
02	IDEX ERRONE	Idexul nu a fost găsit în rezumatul S.G.F. (E.P.)
03	OUV. MF. ACTI	Deschidere simultană a două fișiere ale aceluiași multifișier bandă (E.P.)
04	DVT ABSENT	Nu s-a făcut declararea tipului perifericului (E.P.)
07	DVT:TDF ≠ ASS	Tipul perifericul declarat prin ASSIGN diferă de cel declarat în program (E.P.)
0B	MT:PROTEGEE	Încercare de scriere pe o bandă protejată la scriere
13	ON INCONNU	Opțiunea ON diferită de cea declarată

15	FS:INCONNU	Scrierea unui fișier pe un volum căruia nu îi este asignat
21	FN:INCONNU	Fișierul nu apare pe volum
22	GEN INCONNU	Nr. de generație absent pe bandă
24	CD INCONNU	Data de creare absentă pe volum
25	VN INCONNU	Numărul de versiune absent pe volum
29	F. NON TROUVE	Fișierul nu este găsit pe volum
33	BFS ERRONE	Argumentul BFS declarat în exploatare diferit de BFS declarat la creare
35	F. DEJA CREE	Încercare de creare a unui fișier cu aceleași caracteristici ca ale unui fișier deja existent
38	REFUSAL. NEW	Se cere alocarea NEW pe un volum ocupat
39	F. NON ECRIT	Încercare de deschidere prin citire a unui fișier alocat dar necreat
42	HDR3 ABSENT	Cartea nu a fost găsită în zona partajată (E.P.)
44	F. NON ACTIF	Tentativă de închidere a unui fișier nedeschis (E.P.)
61	TOV SATUREE	Tabela de ocupare a volumului completă
65	AC. FP. ERR	Eroare detectată la verificarea parolei fișierului
71	NO. APP. ABS	Periferic nedisponibil
77	DIM. INSUFF.	Dimensiunea alocată fișierului este insuficientă
78	RCF ERRONE	Formatul articolului diferit la exploatare de cel dat la creare
84	BFS INC. RCS	Mărimea articolului incompatibilă cu dimensiunea zonei tampon (E.P.)
98	RCS ERRONE	Dimensiunea articolului incorectă (E.P.)

Scanned with OKEN Scanner

ERORI DETECTATE DE MODULELE DE ACCES S.G.F.

Nr. de eroare	*Mesaj*	*Natura erorii*
02	E/S ERRONEE	Incident hardware la intrare/ieșire
07	ZP REMPLIE	Nu mai există spațiu disponibil în zona partajată
16	DEPSMT ZONE	Tentativă de violare a zonei la scrierea fișierelor
17	BLOC ABSENT	Blocul curent nu a fost citit
20	PRA ERRONE	Incercare de acces în afara zonei alocate fișierului

BIBLIOGRAFIE

1) — Moniteur SIRIS 3; manuel d'utilisation et d'operation 4002 E3/FR.
2) — Système d'exploitation SIRIS; manuel opérateur 4533 E/FR;
3) — Normes de programmation, manuel d'utilisation 3193 E3/FR.
4) — Programme bibliothécaire, manuel d'utilisation et d'opération 3078 E5/FR.
5) — Editeur de liens 3071 E/FR.
6) — Système de gestion de fichiers 3073 E/FR.
7) — Katzan Harry jr. — Operating systems architecture. Spring Joint Computer Conference 1970.
8) — Denning J. Peter — Third Generation Computer Systems. Computing surveys, vol. 3, nr. 4, 1971.
9) — Munteanu Emil — Utilizarea calculatoarelor în prelucrarea datelor. Ed. Dacia, Cluj-Napoca, 1974.
10) — Preoteasa Petre — Limbaje de programare. Revista de statistică, nr. 2, 1976.
11) — Listeș V. — Sistemul FELIX C-256, Cluj-Napoca, 1974.

Scanned with OKEN Scanner

CUPRINS

Capitolul I ORGANIZAREA GENERALĂ A SISTEMULUI FELIX

— Sistemul de calcul FELIX 9
— Unitatea de memorie 11
— Structura memoriei 13
— Unitatea de comandă 14
— Unităţile de schimburi multiple 25
— Unităţile periferice 26
— Configuraţia unui sistem de calcul 30
— Desfăşurarea operaţiilor de intrare/ieşire 31

Capitolul II SISTEMUL DE OPERARE SIRIS

— Generalităţi 33
— Structura sistemului de operare SIRIS 35
— Lucrare şi fază 36
— Tipul lucrărilor şi gestionarea memoriei 39
— Gestiunea lucrărilor 41
— Fişierele sistemului 42
— Bibliotecile sistemului 45
— Bibliotecarul 47
— Monitorul 47
— Sistemul de gestiune al fişierelor (S.G.F.) 80
— Generatorul de sistem 52

Scanned with OKEN Scanner

Capitolul III DESCRIEREA PRINCIPALELOR CARTELE DE COMANDĂ

— Convenţii de scriere 53
— Formatul cartelelor de comandă 53
— Cartela JOB 54
— Cartela EOJ 54
— Cartela de comentarii 55
— Cartela LINK 55
— Cartela RUN 55
— Cartela COMPILE 57
— Cartela STANDBY 59
— Cartela EOF 60
— Tipuri de rulări simple 60
— Folosirea opţiunii CDC a compilatorului 61

Capitolul IV DESCRIEREA STRUCTURII FIŞIERELOR

— Noţiunea de fişier 65
— Organizarea fişierelor 66
— Organizarea suporturilor 67
— Structura etichetelor standard pentru fişiere pe bandă 68
— Structura etichetelor standard pentru fişiere pe disc 72

Capitolul V EXPLOATAREA FIŞIERELOR ÎN FORTRAN

— Principalele instrucţiuni FORTRAN pentru exploatarea fişierelor 78
— Formatele articolelor 82
— Structura blocului şi articolului în fişiere exploatate secvenţial 83
— Structura blocului şi articolului în fişiere exploatate în acces direct 85
— Declararea fişierelor într-un program FORTRAN .. 86

Scanned with OKEN Scanner

Capitolul VI CARTELE DE COMANDĂ PENTRU FIȘIERE

— Cartela ASSIGN 90
— Cartela LABEL 91
— Cartela INIT 93
— Cartela ALLOC 93
— Cartela DELETE 94
— Exemplul 1 94
— Exemplul 2 97

Capitolul VII PROGRAMUL BIBLIOTECAR

— Descrierea bibliotecilor 101
— Bibliotecarul 103
— Identificarea bibliotecilor și cărților 104
— Generalități asupra cartelelor de comandă ale bibliotecarului 105
— Descrierea cartelelor de comandă ale bibliotecarului 106
— Exemplul 1 114
— Exemplul 2 116

Capitolul VIII SEGMENTAREA PROGRAMELOR

— Generalități despre segmentare 119
— Ordinele SEGMENT și SEGDATA 121
— Cartele de comandă folosite la segmentare 123
— Exemplu 126

Capitolul IX FACILITĂȚI DE RULARE A PROGRAMELOR ÎN SISTEMUL SIRIS

— Compilarea 134
— Editarea legăturilor 136
— Lansarea unui program 137

Scanned with OKEN Scanner

— Exemplu 139
— Proceduri catalogate 143
— Exemplu 146
— Lucrul cu biblioteci de subprograme translatabile .. 147
— Exemplu 149

Capitolul X DEPANAREA PROGRAMELOR

— Erori depistate la compilare 151
— Erori semnalate la editarea legăturilor 153
— Erori care împiedică execuţia 154
— Erori la execuţie 154
— Descrierea listei obiect furnizate de compilatorul FORTRAN 157
— Detectarea instrucţiunii din program care a produs eroarea 161

Capitolul XI PROGRAME UTILITARE

— Programul utilitar de întreţinere (MAINTENANCE) 166
— Programul utilitar de inventariere (INVFICH) 174
— Exemplu 176

Capitolul XII UNELE PROBLEME ACTUALE ALE SISTEMELOR DE OPERARE

— Sisteme de calcul 183
— Caracteristicile unui sistem de operare 185
— Tipuri de sisteme de operare 186
— Direcţii de dezvoltare a tehnicilor de programare .. 190

Scanned with OKEN Scanner

ANEXA 1 Codul EBCDIC 193

ANEXA 2 Program ASSIRIS pentru determinarea timpului 195

ANEXA 3 Coduri de eroare SGF 198

Bibliografie 199

Scanned with OKEN Scanner

10 lei

Lucrarea prezintă facilitățile sistemului de operare, utilizînd limbajul FORTRAN de cea mai largă circulație. Ea se adresează elevilor din clasele speciale de informatică, studenților și în general tuturor utilizatorilor sistemului de calcul FELIX C-256 din centre de calcul și întreprinderi; în consecință este menită să găsească o largă răspîndire, bine meritată pentru eforturile și competența de care fac dovadă cei doi autori, ambii cercetători de certă și recunoscută valoare.

acad. NICOLAE TEODORESCU

Scanned with OKEN Scanner